# First-Order Methods and Principled Frameworks for Optimization



Sofiane Tanji

2026

*Thesis submitted in partial fulfillment of the requirements for the degree of Docteur en sciences de l'ingénieur*

ICTEAM
Louvain School of Engineering
Université catholique de Louvain
Louvain-la-Neuve
Belgium

**Thesis committee:**

| | |
|---|---|
| Pr. François **Glineur** *(Advisor)* | UCLouvain, Belgium |
| Pr. Silvia **Villa** | UniGe, Italy |
| Dr. Mehdi **Madani** | N-SIDE, Belgium |
| Pr. Benoit **Legat** | UCLouvain, Belgium |
| Pr. Mathieu **Besançon** | INRIA Grenoble, France |
| Pr. Pierre-Antoine **Absil** *(Chair)* | UCLouvain, Belgium |

# ABSTRACT

At modern scales, mathematical optimization requires methods whose computational cost remains manageable and that come with working certificates, that is, mathematical guarantees of convergence. This thesis investigates how problem structure and modeling choices can guide the design and selection of first-order methods.

First, we present `snacks`, a new large-scale solver for kernel support vector machines that combines a Nyström approximation of the kernel matrix with a restarted accelerated stochastic subgradient method. On the tested benchmarks, `snacks` is the fastest or tied solver while remaining within 0.9 percentage points of the best test error. Its speed advantage grows with dataset size.

Second, we study first-order methods for computing convex hull prices in electricity markets. These prices are obtained from a partial Lagrangian dual whose objective is a sum of non-smooth convex functions, one per generator. We benchmark a large set of dual first-order methods, most of which had not previously been applied to convex hull pricing. We introduce variable-metric multicut bundle methods, which exploit the separability across generators and are the strongest performers under the time limit imposed by the markets.

Third, we address the choice of a first-order method for a given problem, which amounts to identifying a formulation to which a known convergence theorem applies. We present `Argo.jl`, a disciplined modeling language in Julia that matches problems and their reformulations against a catalogue of convergence theorems. `Argo.jl` returns certified first-order methods and ranks them by worst-case performance whenever their guarantees are comparable.

Finally, in fair learning across sensitive groups, fairness-by-penalization methods often yield Pareto-inefficient models, while the effect of existing Pareto-efficient approaches on a given fairness metric is difficult to control. We introduce Bilevel Adaptive Rescalarization (BADR), which learns the group weights via a bilevel formulation: the lower level enforces Pareto efficiency, while the upper level minimizes a user-specified fairness metric. Our stochastic single-loop algorithm, BADR-SGD, handles the absence of global smoothness and matches the known $\mathcal{O}(\epsilon^{-4})$ lower bound for stochastic bilevel problems. We release `badr`, an open-source Python package implementing BADR, and compare it with various Pareto-efficient

baselines, providing evidence of improved fairness at comparable predictive performance.

Together, these four contributions show that working jointly on the formulation and the methods makes better optimization procedures available, and they provide tools for algorithm selection and fair learning.

# ACKNOWLEDGEMENTS

This thesis consists of four rather different chapters, and tying them together proved harder than expected. The introduction attempts it on formal, scientific grounds, with what success I leave you to judge. In truth, though, these chapters only make sense once you see how they came about and who was there along the way, and this is the story I would like to tell here.

**Committee.** My first thanks go to the members of my thesis committee: to Pierre-Antoine Absil for agreeing to chair it, and to Benoît Legat and Mathieu Besançon for their careful reading of this manuscript and the precision of their reports. I also extend a first thank you to François Glineur and to Silvia Villa and Mehdi Madani, my supervisory committee, who have followed this work from the very beginning and to whom I will return shortly.

**My years in Grenoble.** The idea of pursuing a Ph.D. had taken root a few years earlier, at Ensimag. I had the good fortune to meet a few Ph.D. students there, who made it a point of honour to take the time to guide younger students. Among them are Yassine Laguel and Houssam Zenati, whom I met at the end of my first year, and in whose footsteps I ended up following. Yassine, Houssam, I have been fortunate to have you as mentors, and I am grateful for the time you took to guide me, and for the many discussions we had, which have shaped my path, not only in research but in life. It means a great deal to me that this mentorship, which began so early, has already come full circle, and that I can already count you among my co-authors.

**Intermezzo.** My last year of master's studies led me to Genoa, where I interned and started working with Silvia. There is so much to say about her: I have never met anyone who combines scientific excellence, humility and kindness as she does. However far a talk is from her field, she sits in the front row, taking notes and asking questions. She also has the gift of bringing out the best in everyone. Without her support at a critical time, and her suggestion that I write to François, I would most probably not have pursued a Ph.D. For all of these reasons and many more, *Grazie per tutto, Silvia*.

**Arriving in Louvain-la-Neuve.** I arrived in Louvain-la-Neuve with little background in optimization compared to most applied mathematics students (a single

convex optimization course, taken during Covid), yet François welcomed me into his group. I have always admired how easily he picks up new topics, and how naturally he reasons from first principles. Above all, I am grateful for the academic freedom he gave me. At our very first meeting, he gladly agreed to supervise the rest of my master's thesis, on a topic quite far from his interests; years later, the Ph.D. topic he initially had in mind fills only one of the four chapters of this thesis. François, thank you for your trust, for supporting my own research ideas, even the dubious ones, and for your patience with the slow learner that I am.

**Collaborations.** My Ph.D. was partly funded by a Marie Skłodowska-Curie fellowship within the TraDe-OPT network, which allowed me to spend time in other groups. This is how I came to work with Mehdi Madani on electricity markets, a field entirely new to me. Mehdi, thank you for introducing me to it, and for your sharp eye, which often spotted difficulties weeks before we ran into them. Thank you also for following my work as part of my supervisory committee, and for inviting me to present it at N-SIDE. I also thank Samuel Vaiter for welcoming me to Nice and for sharing his expertise in bilevel optimization, now firmly on my research agenda. Within TraDe-OPT, I thank in particular Yassine Nabou, a great scientist and a great person. I am also grateful to Shuvomoy Das Gupta, also a brilliant mind, whom I met at several conferences and whose warmth and dedication I admire.

**Colleagues.** My office mates were my daily companions for the past five years. Ousmane, your paternal aura made me feel at home from the very first day. Yassine, sharing an office with you allowed for four years of conversations, laughs and a staggering chess record in your favour. Erwan, Pierre, thank you for your kindness and for conversations that were always a pleasure. Rémi, Julien, office mates against your will, thank you for the many breaks and for your friendship. I also thank, in no particular order, Philémon, Charles, Nizar, Teodor, John, Adrien, Matteo, Dâna, Renato, Timothé and Bastien. Among the professors of Louvain-la-Neuve, I thank in particular Geovani Grapiglia, Laurent Jacques and Pierre-Antoine Absil, who each, in their own way, pass on their enthusiasm and energy to younger researchers. I also thank the administrative and technical staff, especially Marie-Christine, Pascale and Dora, whose efficiency is matched only by their kindness. Academia can be a cold and austere world, and I have been fortunate to meet so many good and generous people who made it a warmer place.

**À mes proches.** En dehors de l'université, je pense aussi aux amis rencontrés en chemin, que je considère comme des frères, notamment Nelson, Anas, Ghassen, Cheickna, Kamal et Bilel, merci pour votre amitié et votre soutien. Enfin, et surtout, je veux dire tout mon amour et ma gratitude à ma famille. À mes parents : vous

n'avez jamais ménagé vos efforts et avez tant sacrifié pour que nous puissions faire de bonnes études. Si j'ai accompli quoi que ce soit, je le dois avant tout à votre amour et à votre soutien inconditionnels. À mon frère Anas, qui n'a jamais eu besoin de grands discours pour me montrer la voie : il lui a suffi de donner l'exemple. Ta force et ta droiture ont toujours été un repère pour moi. Merci aussi, Anissa, pour ta gentillesse. Assia, Aya : votre tonton vous aime énormément ! À mes sœurs, Nihal et Lina, qui sont aussi mes meilleures amies. À mes grands-parents, dont l'affection et les prières m'accompagnent depuis l'enfance. À mes beaux-parents, auprès de qui je me suis toujours senti chez moi. Cette dernière année de thèse, particulièrement éprouvante, a pourtant été remplie de joie et de rires grâce à notre fille Soumeya, mon petit rayon de soleil, qui a illuminé mes journées et agité mes nuits. Et à toi, Loujaine : merci pour ton amour, ton soutien et ta patience, qui a été mise à rude épreuve. Tu es une femme exceptionnelle, une chercheuse brillante, et un phare pour nous.

# CONTENTS


1 Introduction 1
- 1.1 Preamble 1
- 1.2 Structure of this thesis 2

2 Background and Preliminaries 11
- 2.1 Optimization Problems 11
- 2.2 Optimization Methods 14
- 2.3 Statistical Learning 15
- 2.4 Notation 18

I First-Order Methods for Large-Scale Nonsmooth Optimization

3 Large-Scale Kernel Support Vector Machines 20
- 3.1 Support vector machines 22
- 3.2 The `snacks` algorithm 28
- 3.3 Computational cost of `snacks` 35
- 3.4 Practical implementation 38
- 3.5 Large-scale numerical experiments 40
- 3.6 Extensions 50
- 3.7 Conclusion 52

4 Dual methods for convex hull pricing 53
- 4.1 Background 55
- 4.2 Unit commitment and convex hull pricing formulations 57
- 4.3 First-order methods 61
- 4.4 Practical implementation 74
- 4.5 Numerical experiments 76
- 4.6 Conclusion 88

II Systematic Frameworks for Optimization

5 Guarantee-based recommendation of first-order methods 91
- 5.1 Introduction 92
- 5.2 A guided tour of `Argo.jl` 97
- 5.3 Problem model and reformulations 105

5.4 Template applicability and theorem certification . . . 111
5.5 Method recommendation from comparable guarantees . . . 114
5.6 Applications and computational examples . . . 119
5.7 Conclusion . . . 125

6 Fairness-informed Pareto optimization 128
6.1 Introduction . . . 130
6.2 Fairness through a Pareto lens . . . 135
6.3 Large-scale methods . . . 141
6.4 Open-source software . . . 149
6.5 Numerical experiments . . . 155
6.6 Conclusion . . . 163
6.A Fairness metrics . . . 164
6.B Convergence analysis . . . 166

7 Conclusions 198
7.1 Open questions . . . 199

Bibliography 201

# 1 INTRODUCTION

## 1.1 PREAMBLE

Mathematical programming (or mathematical optimization) is a principled and scientific approach for decision-making, that is, choosing among available alternatives to achieve a given goal. Formally, decision problems consist in choosing parameters $x$ within a set $\mathcal{C}$ wisely so that an objective function $f : \mathcal{C} \to \mathbb{R}$ is minimized:

$$\min_{x \in \mathcal{C}} f(x) \tag{1.1}$$

Converting a decision-making problem into a mathematical program is referred to as *modeling* in the optimization literature. Once modeled, the second step is to *solve* it. There exists no known universal way (applicable to any $\mathcal{C}$ and $f$) to solve Equation (1.1) in reasonable time. For example, any 3-SAT instance (a problem known to be NP-Hard) can be encoded as an optimization problem. Therefore, a universal polynomial-time solver for Equation (1.1) would imply P = NP. However, a universal method is not needed as long as there exists a method with working certificates for the problem at hand, that is, mathematical guarantees of convergence, such as an upper bound on the error after a given number of iterations. That is what many optimization experts have been concerned with in the last few decades. Numerous optimization methods are now available, each of which is applicable to a specific class of instances of Equation (1.1) under assumptions on $\mathcal{C}$ and $f$. The optimization workflow usually works in the following way: given a concrete mathematical program, a practitioner will try to find a method with working certificates and apply it. The issue is that the form and structure of a mathematical program are not immutable priors; decision-making problems can admit many alternative model formulations, each of which satisfies different assumptions. Therefore, trying to find an appropriate method for a concrete mathematical program without considering its possible reformulations can be shortsighted.

This thesis investigates several settings where working jointly on the formulation and the methods makes better optimization procedures available.

## 1.2 Structure of this thesis

The chapters that follow explore this interplay between problem formulation and optimization methods, first in two concrete applications and then through two broader frameworks. After Chapter 2 introduces the mathematical background used throughout the thesis, the main contributions are organized into two parts. The first part develops first-order methods, which use function values and first-order information (gradients and subgradients), for two large-scale convex nonsmooth optimization problems: training kernel support vector machines (Chapter 3) and computing convex hull prices in electricity markets (Chapter 4). The second part develops frameworks for choosing formulations and methods from convergence guarantees (Chapter 5) and for selecting fair prediction models among Pareto-efficient ones, that is those for which improving one group's performance requires worsening another's (Chapter 6).

The software versions accompanying this thesis are available at https://github.com/SofianeTanji/PhD-Software/.

In the following, we outline the context and main contributions of each chapter. Each chapter is self-contained and may be read independently. Consequently, each begins with a brief introduction, which may partially overlap with the present chapter; the reader is welcome to skip these sections as needed.

### Chapter 3: Large-Scale Kernel Support Vector Machines

**Motivation.** Supervised machine learning aims at developing algorithms that learn a model from labeled examples and predict the labels of new ones. For this, linear models are simple to fit and to interpret and enjoy strong theoretical guarantees, both statistically and computationally. However, such models cannot capture nonlinear dependencies in the data, making them unsuitable for many real-world applications. Kernel methods (Schölkopf & Smola, 2001; Shawe-Taylor & Cristianini, 2004) overcome this issue by mapping the data into a richer space and applying a linear method there. This can be done implicitly without computing the mapping, only through inner products between data points, which the kernel supplies (Aronszajn, 1950). This chapter considers the support vector machine (SVM), introduced in Vapnik (1963) and widely used from the mid-1990s (Steinwart & Christmann, 2008; Vapnik, 1995; Vapnik & Vapnik, 1998), notably for text classification (Joachims, 1999), until deep neural networks superseded it. SVMs remain in use where the model must be small and fast to evaluate as in embedded systems, since its trained model depends only on a subset of the training examples called the support vectors. Example of such systems include hearing aids (GN Hearing AS, 2017) and face recognition on smartphones (Huawei Technologies Co Ltd, 2022).

Training, however, requires the kernel matrix of all pairwise inner products. As this leads to memory complexity quadratic in the number of examples, it is prohibitive at modern scales of millions to billions of examples.

**Problem statement.** Let $(x_i, y_i)$, $i = 1, \dots, n$, be training examples with inputs $x_i$ and labels $y_i \in \{-1, +1\}$. We seek a real-valued function $f$ such that the sign of $f(x)$ predicts the label of the input $x$. A kernel SVM chooses $f$ in a function space $\mathcal{H}$ determined by the chosen kernel, as a solution of

$$\min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i f(x_i)) + \lambda \Omega(f). \tag{1.2}$$

The first term, the hinge loss, measures how well $f$ classifies the training examples. Indeed, it is zero when $f$ classifies $x_i$ correctly with a margin of at least one and grows linearly with the violation otherwise. The second term penalizes the complexity of $f$, and the regularization parameter $\lambda > 0$ balances the two terms. Standard solvers for Equation (1.2) operate on the kernel matrix mentioned above, of size $n \times n$. This chapter develops a solver that avoids this matrix and reduces training time at equal predictive accuracy.

**Related work.** Two main techniques are used to speed up the training of kernel SVMs: kernel approximations and first-order methods. Kernel approximations, such as Nyström subsampling (Williams & Seeger, 2000), replace the full kernel matrix by a low-rank surrogate built from a subset of the training points. This surrogate provides explicit features on which a linear SVM can be trained with any linear solver (Della Vecchia et al., 2024; Yang et al., 2012). State-of-the-art solvers for large-scale kernel SVMs use first-order methods and are of two types, depending on the formulation tackled. On the primal side, the reference solver is Pegasos (Shalev-Shwartz et al., 2011), a stochastic subgradient method applied directly to the nonsmooth objective Equation (1.2). On the dual side, when the regularizer is the squared norm, Equation (1.2) is equivalent to a quadratic program with one variable per training example, which the widely used packages LibSVM (Chang & Lin, 2011) and LibLinear (Fan et al., 2008) solve by coordinate descent. Existing large-scale pipelines pair Nyström features with such generic linear solvers (Pedregosa et al., 2011).

**Contributions.**

- **We introduce `snacks`, a large-scale solver for kernel support vector machines.** It combines a Nyström embedding with the restarted accelerated

stochastic subgradient method of Xu et al. (2019), using fixed optimizer settings across datasets.

- **We analyze the computational cost of `snacks` and describe extensions to other losses and regularizers.** The analysis accounts for feature construction, optimization, and prediction, and relates the optimization cost to the requested accuracy. The extensions require changing only the stochastic oracle, while retaining the embedding and optimizer.
- **We release `snacks` as an open-source Python package with a `scikit-learn` interface.**
- **We conduct extensive numerical experiments and isolate the contribution of the accelerated optimizer.** On the tested benchmarks, `snacks` is the fastest or tied solver while remaining within 0.9 percentage points of the best test error. Its speed advantage grows with dataset size, and the experiments support using fixed optimizer settings without per-dataset tuning.

### Chapter 4: Dual methods for convex hull pricing

**Motivation.** In electricity markets, a system operator determines which generators should run and how much electricity each should produce so that generation meets demand at the lowest overall cost, subject to technical constraints. This scheduling problem is known as *unit commitment*. Decisions such as whether a generator is on or off require discrete variables, making the problem non-convex. This non-convexity also complicates pricing: there may be no uniform electricity prices that make the assigned schedule profit-maximizing for every generator (Scarf, 1994). For example, if a generator cannot recover its start-up and operating costs from electricity revenues, it may prefer to remain off rather than follow the assigned schedule. More generally, even if a generator earns a positive profit, it may benefit from choosing a different feasible schedule, a situation known as *lost opportunity*. This leads markets to supplement electricity revenues with side payments, also called *uplifts*, to compensate generators for following the assigned schedule.

**Problem statement.** For a given market schedule and a proposed set of prices, a generator's *lost opportunity cost* is the difference between the largest profit it could earn by choosing its own feasible schedule and the profit it earns by following the assigned schedule. Convex hull (CH) prices are defined as the prices that minimize the sum of these lost opportunity costs over generators (Gribik et al., 2007; Hogan & Ring, 2003; Ring, 1995). It was first observed in Gribik et al. (2007) that computing CH prices is equivalent to solving the Lagrangian dual associated with the unit commitment problem with dualized power-balance constraints. Equivalently, we solve

$$\min_{\pi \in Q} f(\pi), \quad f(\pi) = \sum_{g \in \mathcal{G}} \max_{x^g \in \mathcal{P}^g} \left[\pi^\top p^g - C^g(x^g)\right] + \max_{0 \leqslant l \leqslant L} \left[-\pi^\top l - C_{VOLL} \sum_{t=1}^{T} (L_t - l_t)\right]. \tag{1.3}$$

Here $\mathcal{G}$ is the set of generators, $x^g = (p^g, u^g, v^g, w^g)$ is a generator schedule, $p^g$ its power output, and $C^g(x^g)$ its total cost. The feasible set $\mathcal{P}^g$ imposes binary commitment, capacity limits, minimum up and down times, and ramp constraints. The vectors $\pi$, $L$, and $l$ contain prices, demand, and served demand over $T$ periods; $Q$ specifies the admissible price bounds and $C_{VOLL}$ is the unit cost of non-served energy. The objective is nonsmooth and convex, and computing a subgradient requires solving one mixed-integer linear program (MILP) per generator. As solving MILPs at each iteration is expensive, the choice of the method is decisive at the scale of real markets.

**Related work.** State-of-the-art methods for computing CH prices are either primal or dual. Primal approaches directly tackle the unit commitment problem and compute CH prices from the optimal dual variables of the power-balance constraints. Because the problem is typically intractable at the scale of real markets, primal methods rely on decomposition techniques that solve the problem incrementally (Andrianesis et al., 2021; Knueven et al., 2022; Paquet, 2021; Yu et al., 2020). Dual approaches instead optimize directly in price space, using the equivalence established in Gribik et al. (2007). The subgradient method was applied to CH pricing in Ito et al. (2013). More recently, Stevens & Papavasiliou (2022) applied a bundle level method and reported that this approach is competitive on real-world market instances. To the best of our knowledge, no other dual first-order methods had been investigated for CH pricing before our work.

**Contributions.**

- **We study a broad family of dual first-order methods for computing CH prices.** Most of these methods had not previously been applied to convex hull pricing.
- **We introduce variable-metric multicut bundle methods that exploit the separability across generators.** They maintain a separate polyhedral approximation for each group of generators and use an adaptive diagonal metric.
- **We release `ConvexHullPricing.jl`, a Julia package for convex hull pricing.** It implements primal and dual methods, provides a common framework for benchmarking them, and computes market quantities such as schedules and uplifts.

- **We compare the methods on 24 large-scale Belgian and Californian market instances.** The variable-metric bundle variants are the strongest performers under the 15-minute time limit, while on the Californian instances subgradient and smoothing-based methods reach the target accuracy about as fast. Ablation experiments quantify the gains from multicut approximations and from the variable metric.

### Chapter 5: Guarantee-based recommendation of first-order methods

**Motivation.** A wide range of first-order methods is now available in the optimization literature. Published algorithms are usually shown to apply to given *templates* of optimization problems, and come with proofs characterizing their convergence rates. A template may describe the objective as a single function or as a sum of functions with different properties, such as smoothness or convexity. To apply a method, the practitioner must therefore check that the problem matches the template covered by its convergence theorem. For a given problem, it is not always easy to identify which templates it can match, especially when considering equivalent reformulations. For example, regrouping the terms of an objective can change the properties of each part without changing the function being minimized. Choosing a method thus comes down to finding a formulation that some existing theorem already covers, which makes the task tedious for the practitioner. Given a specific formulation of an optimization problem, identifying formulations amenable to known convergence results typically requires applying a sequence of transformations until one of the recognized templates is reached. This reformulation process is both time-consuming and error-prone, and its effectiveness depends on the transformations and templates known to the user. Finally, even when several methods apply, their guarantees may concern different measures of error and cannot always be compared directly.

**Problem statement.** We consider structured optimization problems, such as

$$\min_{x \in \mathbb{R}^d} F(x), \quad F(x) = \sum_{i=1}^{m} f_i(x), \tag{1.4}$$

where each term $f_i$ is described by its known properties and available *oracles*, such as routines for computing its value or gradient. Our goal is to identify first-order methods that apply to Equation (1.4), possibly after reformulation of the problem, and rank them by worst-case convergence guarantees whenever these guarantees are comparable. Guarantees are typically comparable when they rely on the same initial information and upper bound the same measure of error, such as the gradient norm or the distance to the set of optimal solutions.

**Related work.** Algebraic modeling languages allow users to describe optimization problems mathematically and automatically transform them into a standard form accepted by a solver (Diamond & Boyd, 2016; Grant & Boyd, 2014; Lubin et al., 2023). TFOCS follows the same principle for first-order methods, with reformulations leading to a prescribed problem form (Becker et al., 2011). In these frameworks, reformulation serves to make a solver applicable. This contrasts with our setting where reformulations are used to find applicable convergence analyses, whose guarantees can then be compared across formulations and methods. These analyses can themselves be automated using the performance estimation framework, which computes worst-case bounds for a given method, problem class, and performance measure (Drori & Teboulle, 2014; Taylor et al., 2017a).

**Contributions.**

- **We introduce a disciplined modeling framework for first-order methods.** It represents optimization problems through their structural assumptions and available oracles, together with admissible reformulations, method templates, and convergence certificates.
- **We extract from the literature a catalogue of convergence theorems for classical first-order methods.** For each theorem, we record the problem class, assumptions, required oracles, performance measure, convergence rate, and bibliographic source, turning it into a reusable template.
- **We design a procedure that matches problems and their reformulations against these templates.** By tracking which guarantees each reformulation step can transfer, we establish whether and how a certificate obtained for a reformulated problem applies to the original problem.
- **We return certified first-order methods and rank them by worst-case performance whenever their guarantees are comparable.**
- **We extend the framework beyond the recommendation of a single method.** We select parameters in exact reformulations and switching times in hybrid methods by minimizing the resulting worst-case bounds. We also recursively certify methods for solving proximal subproblems to the accuracy required by an outer method.
- **We release `Argo.jl`, an open-source Julia package implementing the complete procedure.**

### Chapter 6: Fairness-informed Pareto optimization

**Motivation.** Data-driven decision systems are now commonly used in high-stakes domains such as employment, education, law enforcement, and health care (Citron & Pasquale, 2014). A model with good average predictive performance can

nevertheless perform unevenly across groups defined by sensitive attributes, such as gender or race. To quantify such disparities, *unfairness metrics* assign a numerical value to violations of a chosen fairness criterion, which ask that predictions be independent of sensitive attributes in some sense. Several such metrics exist and are not equivalent: a model may be fair according to one metric but unfair according to another. Reducing these measures of unfairness can, however, come at a cost in predictive performance for some groups. A model with a good fairness score may nevertheless allow the performance of one group to be improved without degrading that of any other group. Our aim is therefore to improve fairness while ensuring that the resulting model admits no such improvement.

**Problem statement.** We consider training data $(x_i, y_i, a_i)$, $i = 1, ..., n$, where $x_i$ is the input, $y_i$ the target output, and $a_i \in \{1, ..., S\}$ the sensitive group of individual $i$. For a predictor $f(w, \cdot)$ with parameters $w \in \mathbb{R}^d$, we define the group losses

$$F_a(w) = \frac{1}{n_a} \sum_{i:a_i=a} \ell(f(w, x_i), y_i) + R(w), \quad a = 1, ..., S, \tag{1.5}$$

where $n_a$ is the number of training examples in group $a$, $\ell$ measures prediction error, and $R$ is an optional regularization term. Considering all group losses leads to the multi-objective problem

$$\min_{w \in \mathbb{R}^d} [F_1(w), ..., F_S(w)]. \tag{1.6}$$

A model is *Pareto-efficient* if no group loss can be decreased without increasing at least one other (Kuhn & Tucker, 1951). The group-loss vectors of these models form the *Pareto front*. Our aim is to find, among these models, one that minimizes a chosen unfairness metric $\Phi$. We learn the group weights through the bilevel problem

$$\begin{aligned} \min_{\lambda \in \Delta_S} \quad & \Phi(w^*(\lambda), \lambda) \\ \text{subject to} \quad & w^*(\lambda) \in \arg\min_{w \in \mathbb{R}^d} \sum_{a=1}^{S} \lambda_a F_a(w), \end{aligned} \tag{1.7}$$

where $\Delta_S$ is the probability simplex. The lower level produces a Pareto-efficient model whenever its minimizer is unique. For convex group losses, this uniqueness is guaranteed if at least one strongly convex loss has a positive weight. The upper level then chooses the weights to minimize the model's unfairness.

**Related work.** A common family of approaches, known as *in-processing*, enforces fairness during training, most often through fairness-promoting regularization (Do et al., 2022; Donini et al., 2018; Zafar, Valera, Gomez Rodriguez, et al., 2017). Such regularization adds a penalty for unfairness to the training objective, often yielding Pareto-inefficient models, as illustrated in Figure 1.1.

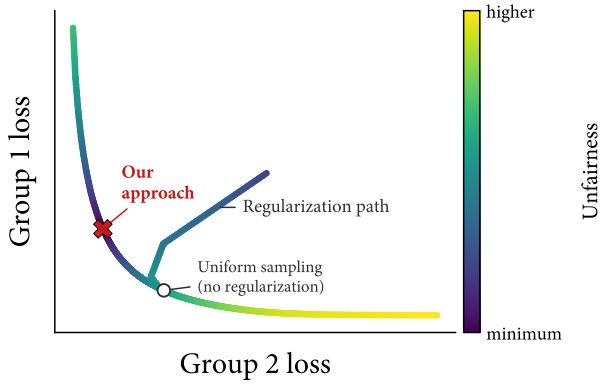



Figure 1.1: Illustration of the impact of fairness regularization on Pareto efficiency in the objective space. Models on the regularization path can be Pareto-inefficient.

A standard way to obtain Pareto-efficient models is weighted-sum *scalarization*, which combines the group losses into a single weighted objective. When all weights are strictly positive, every exact minimizer of this objective is Pareto-efficient. Existing approaches can be interpreted as selecting a particular fixed weighting of the group losses, either explicitly (Sivarajkumar et al., 2023) or implicitly, as in minimax fairness (Hashimoto et al., 2018; Martinez et al., 2020). These choices are oblivious to any unfairness metric, which make such approaches hard to align with stakeholders' fairness priorities. Related bilevel approaches allow weights to depend on both group and class (Ozdayi et al., 2021; Roh et al., 2020). In that case, the training objective is not necessarily a weighted sum of group losses, so Pareto efficiency with respect to those losses is no longer guaranteed.

**Contributions.**

- **We introduce Bilevel Adaptive Rescalarization (BADR) for fair learning.** BADR learns group weights through a bilevel formulation to minimize a user-specified unfairness metric subject to groupwise Pareto efficiency. The lower level enforces efficiency, while the upper level minimizes the chosen unfairness criterion.

- **We develop two large-scale, single-loop bilevel algorithms and establish their convergence guarantees.** BADR-GD and BADR-SGD jointly update the model and group weights in the deterministic and stochastic settings, respectively. We establish convergence rates of $\mathcal{O}(1/T)$ and $\mathcal{O}\left(1/\sqrt{T}\right)$ after $T$ iterations, using a measure that combines upper-level stationarity with lower-level error. Notably, the analysis handles the absence of a common assumption in the bilevel literature which doesn't hold for our setting, and yields results that apply beyond the fairness setting.
- **We release `badr`, an open-source Python package implementing BADR.** It follows the `scikit-learn` interface, provides seven unfairness metrics, and supports custom metrics and standard learning models. We also compare the available optimization methods through performance profiles and scalability experiments.
- **We conduct extensive numerical experiments on real-world datasets and seven fairness metrics.** We compare `badr` with various fixed weighting schemes that are common Pareto-efficient baselines, namely : uniform sampling, balanced sampling, minimax fairness, and one-group fitting. The experiments provide evidence of improved fairness at comparable predictive performance, while preserving groupwise Pareto efficiency. Visualizations for two and three groups show how unfairness varies along the Pareto front and motivate optimizing over it.

# 2 BACKGROUND AND PRELIMINARIES

This chapter collects the definitions and standard results from optimization and statistical learning that the formulations and algorithms of later chapters use.

## 2.1 OPTIMIZATION PROBLEMS

### 2.1.1 Mathematical programs

Let $\mathcal{E}$ be a finite-dimensional normed vector space with dual space $\mathcal{E}^*$. We consider problems of the form

$$\min_{x \in \mathcal{C}} f(x), \tag{2.1}$$

where $\mathcal{C} \subseteq \mathcal{E}$ and $f : \mathcal{E} \to \mathbb{R} \cup \{+\infty\}$ is proper, with $f$ finite at some point of $\mathcal{C}$. A feasible point belongs to $\mathcal{C}$ and has finite objective value. The optimal value and solution set are

$$f_* = \inf_{x \in \mathcal{C}} f(x), \quad X^* = \{x \in \mathcal{C} : f(x) = f_*\}. \tag{2.2}$$

A point $x^* \in X^*$ is a global minimizer.

### 2.1.2 Convex structure

**Definition 2.1.** (Convex set) A set $\mathcal{C} \subseteq \mathcal{E}$ is convex if

$$\lambda x + (1-\lambda) y \in \mathcal{C} \quad \forall x, y \in \mathcal{C}, \forall \lambda \in [0,1]. \tag{2.3}$$

For an extended real-valued function $f : \mathcal{E} \to \mathbb{R} \cup \{+\infty\}$, its effective domain and epigraph are

$$\operatorname{dom} f = \{x \in \mathcal{E} : f(x) < +\infty\}, \quad \operatorname{epi} f = \{(x,t) \in \mathcal{E} \times \mathbb{R} : f(x) \leqslant t\}. \tag{2.4}$$

**Definition 2.2.** (Convex function) A function $f : \mathcal{E} \to \mathbb{R} \cup \{+\infty\}$ is convex if $\operatorname{dom} f$ is convex and

$$f(\lambda x + (1-\lambda) y) \leqslant \lambda f(x) + (1-\lambda) f(y) \quad \forall x, y \in \operatorname{dom} f, \forall \lambda \in [0,1]. \tag{2.5}$$

Equivalently, $f$ is convex if and only if $\operatorname{epi} f$ is convex.

### 2.1.3 Subgradients and optimality conditions

**Definition 2.3.** (Subgradient and subdifferential) Let $f : \mathcal{E} \to \mathbb{R} \cup \{+\infty\}$ be proper and convex. A vector $g \in \mathcal{E}^*$ is a *subgradient* of $f$ at $x \in \operatorname{dom} f$ if

$$f(y) \geqslant f(x) + \langle g, y - x \rangle \quad \forall y \in \mathcal{E}. \tag{2.6}$$

The set of all subgradients at $x$ is the *subdifferential* $\partial f(x)$.

If $f$ is differentiable at $x$, then $\partial f(x) = \{\nabla f(x)\}$. Moreover, $x^*$ minimizes a proper convex function over $\mathcal{E}$ if and only if $0 \in \partial f(x^*)$. For a convex set $\mathcal{C}$ and $x \in \mathcal{C}$, define the normal cone

$$N_{\mathcal{C}}(x) = \{g \in \mathcal{E}^* : \langle g, y - x \rangle \leqslant 0 \quad \forall y \in \mathcal{C}\}. \tag{2.7}$$

When the subdifferential sum rule applies, $x^*$ minimizes $f$ over $\mathcal{C}$ if and only if

$$0 \in \partial f(x^*) + N_{\mathcal{C}}(x^*). \tag{2.8}$$

### 2.1.4 Lipschitz continuity, smoothness, and strong convexity

Throughout this subsection, $\mathcal{E} = \mathbb{R}^d$ with the Euclidean norm and inner product.

**Definition 2.4.** (Lipschitz continuity) A function $f : \mathcal{E} \to \mathbb{R}$ is $M$-*Lipschitz* on $\mathcal{C} \subseteq \mathcal{E}$ if

$$|f(x) - f(y)| \leqslant M\|x - y\| \quad \forall x, y \in \mathcal{C}. \tag{2.9}$$

For a convex function on an open convex set, $M$-Lipschitz continuity is equivalent to $\|g\| \leqslant M$ for every $g \in \partial f(x)$ and every $x$ in that set.

**Definition 2.5.** (Smoothness) A differentiable function $f : \mathcal{E} \to \mathbb{R}$ is $L$-*smooth* if

$$\|\nabla f(x) - \nabla f(y)\| \leqslant L\|x - y\| \quad \forall x, y \in \mathcal{E}. \tag{2.10}$$

**Lemma 2.6.** (Descent lemma) *If $f : \mathbb{R}^d \to \mathbb{R}$ is $L$-smooth, then*

$$f(y) \leqslant f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|y - x\|^2 \quad \forall x, y \in \mathbb{R}^d. \tag{2.11}$$

**Definition 2.7.** (Strong convexity) A proper function $f : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ is $\mu$-*strongly convex* ($\mu \geqslant 0$) if $f - \frac{\mu}{2}\|\cdot\|^2$ is convex. In that case,

$$f(y) \geqslant f(x) + \langle g, y - x \rangle + \frac{\mu}{2}\|y - x\|^2 \tag{2.12}$$

for all $x, y \in \operatorname{dom} f$ and $g \in \partial f(x)$.

For a function that is both $L$-smooth and $\mu$-strongly convex with $\mu > 0$, $\kappa = L/\mu$ is the condition number.

### 2.1.5 Proximal operators and projections

**Definition 2.8.** (Proximal operator) Let $g : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ be proper, closed, and convex, and let $\gamma > 0$. Its *proximal operator* is

$$\operatorname{prox}_{\gamma g}(x) = \operatorname*{arg\,min}_{y \in \mathbb{R}^d} \left\{ g(y) + \frac{1}{2\gamma}\|y - x\|^2 \right\}. \tag{2.13}$$

The minimizer is unique in the Euclidean setting (Beck, 2017). Projection onto a nonempty closed convex set $\mathcal{C}$ is the special case $\operatorname{Proj}_{\mathcal{C}} = \operatorname{prox}_{\gamma \iota_{\mathcal{C}}}$, where $\iota_{\mathcal{C}}$ is $0$ on $\mathcal{C}$ and $+\infty$ elsewhere. A function is *proximable* when this operator is available in closed form or can be evaluated cheaply. These operators are used in Chapter 4 and Chapter 5.

### 2.1.6 Duality

**Definition 2.9.** (Convex conjugate) The *convex conjugate* of $f : \mathcal{E} \to \mathbb{R} \cup \{+\infty\}$ is

$$f^*(s) = \sup_{x \in \mathcal{E}} \{\langle s, x\rangle - f(x)\}, \quad s \in \mathcal{E}^*. \tag{2.14}$$

The conjugate is closed and convex. The Fenchel–Young inequality

$$f(x) + f^*(s) \geqslant \langle s, x\rangle \tag{2.15}$$

holds for all $x$ and $s$, with equality if and only if $s \in \partial f(x)$. If $f$ is proper, closed, and convex, then $f^{**} = f$ (Rockafellar, 1970).

Consider the conic problem

$$\inf_{x \in \mathcal{E}} f(x) \quad \text{subject to} \quad c(x) \in -\mathcal{K}, \tag{2.16}$$

where $\mathcal{K}$ is a closed convex cone with dual cone

$$\mathcal{K}^* = \{\lambda : \langle \lambda, z\rangle \geqslant 0 \quad \forall z \in \mathcal{K}\}. \tag{2.17}$$

For $\lambda \in \mathcal{K}^*$, define

$$\mathcal{L}(x, \lambda) = f(x) + \langle \lambda, c(x)\rangle, \quad d(\lambda) = \inf_x \mathcal{L}(x, \lambda). \tag{2.18}$$

The dual problem is $\sup_{\lambda \in \mathcal{K}^*} d(\lambda)$, and weak duality gives

$$\sup_{\lambda \in \mathcal{K}^*} d(\lambda) \leqslant \inf_{x : c(x) \in -\mathcal{K}} f(x). \tag{2.19}$$

Under an appropriate constraint qualification, such as Slater's condition for convex conic problems, equality holds (Boyd & Vandenberghe, 2004). Chapter 4 builds on this Lagrangian structure to recover convex-hull prices.

### 2.1.7 Equivalent formulations, relaxations, and surrogates

We call two formulations *equivalent* when their solutions can be mapped to one another at negligible cost, as for the primal and dual support-vector-machine formulations in Chapter 3. For a minimization problem, a *relaxation* enlarges the feasible set or lowers the objective and therefore gives a lower bound; convex relaxation is central to Chapter 4. A *surrogate* replaces an objective by a more tractable one, for example by smoothing or scalarization. In Chapter 5, reformulations are treated as a collection to be matched with compatible algorithms.

## 2.2 Optimization Methods

### 2.2.1 Oracles and performance criteria

An *oracle* returns specified information when queried at a point.

We consider here standard black-box oracles: a zeroth-order oracle returns a function value $f(x)$; a first-order oracle returns a subgradient in $\partial f(x)$; a proximal oracle returns $\operatorname{prox}_{\gamma f}(x)$; and a linear-minimization oracle returns $\arg\min_{y \in \mathcal{C}} \langle s, y \rangle$ (Nemirovskii, 1983). We can also consider inexact oracles, which return the oracle information up to a controlled error.

We focus on the setting where the oracle call is the most expensive operation. Therefore, complexity is measured by the number of oracle calls required to reach an accuracy $\epsilon$. Relevant accuracy criteria are objective suboptimality $f(x) - f_*$, distance to the solution set, stationarity measures such as $\inf_{g \in \partial f(x)} \|g\|$, and primal–dual gaps. An important element for optimization methods is knowing when to stop. Practical stopping rules usually use computable versions of the quantities above or a given oracle-call budget.

### 2.2.2 Problem classes

An *optimization problem class* is a family of problems characterized by common structural assumptions on the objective and feasible set, together with the oracle information available to a method. These characteristics determine which algorithms apply and which convergence guarantees can be invoked.

We describe three recurrent classes. In *smooth convex optimization*, $f$ is convex and L-smooth and is accessed through gradients. In *composite optimization*, $\min_x f(x) + g(x)$ combines a smooth convex $f$ with a proper, closed, convex,

and proximable function $g$. In *constrained optimization*, $\min_{x \in \mathcal{C}} f(x)$ is handled through projections or, when only a linear-minimization oracle is available, conditional gradient methods.

Matching a formulation to such a class is the problem we partly automate in Chapter 5[1].

### 2.2.3 First-order methods

First-order methods are iterative schemes building a sequence of candidate solutions using gradient or subgradient information returned by an oracle. A simple example is *gradient descent*, which moves the current iterate in the direction opposite to the gradient:

$$x_{k+1} = x_k - \gamma_k \nabla f(x_k). \tag{2.20}$$

For the standard L-smooth convex problem class, a constant step size $\gamma_k = 1/L$ suffices to guarantee a convergence rate of $\mathcal{O}(1/k)$ on the objective suboptimality. More sophisticated optimization methods can accelerate this rate up to $\mathcal{O}\left(1/k^2\right)$ (Nesterov, 2004; Nesterov, 1983).

In the setting where the function is nonsmooth but Lipschitz, we can still minimize the function with *subgradient* information, replacing the gradient by $g_k \in \partial f(x_k)$:

$$x_{k+1} = x_k - \gamma_k g_k. \tag{2.21}$$

Under standard assumptions and an appropriate step-size or averaging scheme, it achieves the characteristic $\mathcal{O}\left(1/\sqrt{k}\right)$ rate (Boyd et al., 2003). We study similar problems in both Chapter 3 and Chapter 4.

Whenever the objective function exhibits a finite sum structure $f = \frac{1}{n}\sum_{i=1}^{n} f_i$, it is possible to leverage stochastic methods, which use a randomly sampled component (sub)gradient instead of evaluating all $n$ components at every iteration. Stochastic methods are what allows scalable algorithms in Chapter 3 and Chapter 6.

## 2.3 Statistical learning

The notation below is used in Chapter 3 and Chapter 6; see Mohri et al. (2012) for a general treatment.

[1] In Chapter 5, problem classes are called *templates* when paired with contextual information about which oracles are available.

### 2.3.1 Learning problem and risk

Let $\mathcal{X}$ and $\mathcal{Y}$ be input and output spaces, and let $\mathbb{P}$ be an unknown distribution on $\mathcal{X} \times \mathcal{Y}$. A score function $h : \mathcal{X} \to \mathbb{R}$ belongs to a hypothesis space $\mathcal{H}$, and its performance under a loss $\ell$ is measured by the *risk*

$$\mathcal{R}(h) = \mathbb{E}_{(x,y)\sim\mathbb{P}}[\ell(h(x), y)]. \tag{2.22}$$

Given an i.i.d. sample $(x_i, y_i)_{i=1}^n$, the *empirical risk* is

$$\mathcal{R}_n(h) = \frac{1}{n}\sum_{i=1}^{n} \ell(h(x_i), y_i). \tag{2.23}$$

### 2.3.2 Empirical risk minimization and regularization

For a parametrization $h = h_w$, empirical risk minimization solves

$$\min_{w\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} \ell(h_w(x_i), y_i). \tag{2.24}$$

Typical losses are the squared loss $\ell(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$, the hinge loss $\ell(\hat{y}, y) = \max(0, 1 - y\hat{y})$, and the logistic loss $\ell(\hat{y}, y) = \log\left(1 + e^{-y\hat{y}}\right)$. The hinge loss is used in Chapter 3.

Regularization controls the complexity of the learned predictor:

$$\min_{w\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} \ell(h_w(x_i), y_i) + \lambda R(w), \tag{2.25}$$

where $\lambda > 0$. Common choices for the regularizer are the squared $\ell_2$ norm, the $\ell_1$ norm, and the elastic net, used in Chapter 5. The examples listed above are all proximable (their proximal operators are available in closed form), which is a structure many first-order solvers exploit to achieve faster convergence rates than for general nonsmooth problems.

### 2.3.3 Linear and kernelized prediction

For $x \in \mathbb{R}^d$, the simplest hypothesis class scores inputs by a single inner product,

$$h_w(x) = \langle w, x\rangle. \tag{2.26}$$

Changing the loss function then gives different linear predictors: linear regression for the squared loss, logistic regression for the logistic loss, and a linear support vector machine for the hinge loss.

Kernel methods work with linear predictors but replace $x$ by its image under a feature map $\Phi : \mathcal{X} \to \mathcal{H}$. This map sends each input to a reproducing kernel Hilbert space. The score becomes $h_w(x) = \langle w, \Phi(x) \rangle_{\mathcal{H}}$. As the associated kernel

$$k(x, x') = \langle \Phi(x), \Phi(x') \rangle_{\mathcal{H}} \tag{2.27}$$

returns the inner product needed for linear predictors, kernel methods can be implemented without explicitly constructing the features (Schölkopf & Smola, 2001; Shawe-Taylor & Cristianini, 2004).

Standard kernel solvers require a $n \times n$ kernel matrix. Storing and manipulating this matrix quickly becomes too expensive as the sample size $n$ grows. We discuss this scalability issue and how to bypass it in Chapter 3.

## 2.4 NOTATION

Vectors and matrices are denoted by lower-case and upper-case letters, respectively. Unless stated otherwise, $\langle \cdot, \cdot \rangle$ is the Euclidean inner product and $\|\cdot\|$ its associated norm.

| Symbol | Meaning |
|---|---|
| $\overline{\mathbb{R}} = \mathbb{R} \cup \{+\infty\}$ | upper extended real line |
| $\langle x, y \rangle$ | (Euclidean) inner product |
| $\|\cdot\|$ | associated norm |
| $\mathcal{C}$ | feasible set |
| $\mathcal{X}, \mathcal{Y}$ | input and output spaces |
| $\partial f(x)$ | subdifferential of $f$ at $x$ |
| $\nabla f$ | gradient of $f$ |
| $f_*$ | optimal value |
| $\mathrm{Proj}_{\mathcal{C}}$ | Euclidean projection onto $\mathcal{C}$ |
| $\mathcal{O}(\cdot)$ | asymptotic (big-$\mathcal{O}$) order |

Part I

# First-Order Methods for Large-Scale Nonsmooth Optimization

# 3 LARGE-SCALE KERNEL SUPPORT VECTOR MACHINES


## ABSTRACT

In this chapter, we focus on faster kernel SVM solvers that scale to modern, large-scale datasets.

Kernel methods provide a powerful framework for non-parametric learning. They are based on kernel functions and allow learning in a rich functional space while applying well-known linear statistical learning tools, such as ridge regression or support vector machines. In their standard form, kernel methods store the kernel matrix, which entails quadratic memory costs in the number of data points and becomes prohibitive at modern scales, with millions to billions of examples.

As the first contribution of this thesis, we present `snacks`[2], a new large-scale solver for kernel support vector machines (SVMs). `snacks` combines a Nyström approximation of the kernel matrix with an accelerated stochastic subgradient method. Through an extensive empirical evaluation, we show that it is competitive with existing SVM solvers on a range of benchmark datasets.


## BASED ON

*The developments presented in this chapter are based on the following publication:*

- *Tanji, S., Della Vecchia, A., Glineur, F., & Villa, S. (2023). Snacks: a fast large-scale kernel SVM solver.* 2023 European Control Conference (ECC), *1–6.*

*This work started during my internship at MaLGa in Genoa, Italy under the supervision of Professors Silvia Villa and Lorenzo Rosasco. I completed the work during my PhD with the additional supervision of Professor François Glineur. For this thesis, the implementation of* `snacks` *was revised and all the numerical experiments of Section 3.5 were rerun with an updated protocol and set of baselines, so the results reported here differ from those of the original publication.*

[2] https://github.com/SofianeTanji/PhD-Software/Snacks

### IN THIS CHAPTER

The first section of this chapter (Section 3.1) is dedicated to a review of support vector machines, their existing formulations and learning algorithms. In Section 3.2, we derive `snacks`, our main contribution, step by step, and report computational complexity results under classical assumptions in Section 3.3. Section 3.4 presents the practical implementation of `snacks` and details about the open-source library. Section 3.5 is dedicated to detailed numerical experiments on various datasets, showing that `snacks` is competitive with state-of-the-art solvers on several medium to large-scale classification tasks. In Section 3.6, we discuss specific extensions of `snacks` to other settings, such as alternative losses and regularizers, for which only the stochastic oracle changes while the embedding and optimizer are unaffected. We also highlight specific settings in which faster convergence can be achieved. Finally, Section 3.7 concludes the chapter and discusses possible future work.

## 3.1 SUPPORT VECTOR MACHINES

In this section, we introduce the kernel SVM problem and present both primal and dual formulations as well as existing learning methods for solving them.

### 3.1.1 Problem statement

To keep the chapter self-contained, we briefly recall the key elements required to state the kernel SVM problem.

**From statistical learning to regularized ERM.** In supervised learning, the data are modeled by a pair of random variables $(X, Y)$ taking values in $\mathcal{X}$ (the input space) and $\mathcal{Y} \subset \mathbb{R}$ (the output space), with joint distribution $P$. Given a loss function $\ell : \mathcal{Y} \times \mathbb{R} \to \mathbb{R}$, the goal is to minimize the expected risk

$$f^* \in \arg\min_{f \in \mathcal{F}} L(f) \qquad L(f) \stackrel{\text{def}}{=} \mathbb{E}_{(X,Y)\sim P}[\ell(Y, f(X))], \tag{3.1}$$

where $\mathcal{F}$ denotes the set of all measurable functions from $\mathcal{X}$ to $\mathbb{R}$. Since $P$ is unknown, we rely on a training set $S_n = \{(x_i, y_i)\}_{i=1}^n$ of i.i.d. samples from $P$ and consider the empirical risk

$$\hat{L}_n(f) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(x_i)). \tag{3.2}$$

Minimizing $\hat{L}_n$ over all measurable functions is ill-posed, as infinitely many functions can interpolate the data and achieve zero empirical risk. A standard remedy is to restrict to a hypothesis space $\mathcal{H} \subset \mathcal{F}$ and to add a regularizer $\Omega$ that penalizes functional complexity, leading to the regularized empirical risk minimization (ERM) problem

$$\hat{f}_n \in \arg\min_{f \in \mathcal{H}} \hat{L}_n(f) + \lambda \Omega(f), \tag{3.3}$$

where $\lambda > 0$ is a regularization parameter.

**Kernel hypothesis space and representer theorem.** We restrict attention to the classification setting, where $\mathcal{Y}$ is finite. In what follows, we further specialize to binary classification with $\mathcal{Y} = \{-1, +1\}$, since multiclass problems (that is, with $\mathcal{Y} = \{1, ..., C\}$) can be handled by standard reductions to a sequence of binary classification tasks.

In kernel methods, $\mathcal{H}$ is taken to be a reproducing kernel Hilbert space (RKHS) associated with a symmetric positive definite kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Let $k_x \stackrel{\text{def}}{=}$

$k(x, \cdot)$ and let $\|\cdot\|_{\mathcal{H}}$ denote the RKHS norm. Throughout, we assume that the regularizer depends on the RKHS norm, i.e., with a slight abuse of notation,

$$\Omega(f) = \Omega\big(\|f\|_{\mathcal{H}}\big), \tag{3.4}$$

where $\Omega : \mathbb{R}_+ \to \overline{\mathbb{R}}$ is proper, lower semicontinuous and nondecreasing.

**Theorem 3.1.** (Representer theorem, Schölkopf et al., 2001) *If (3.3) has a minimizer, then it has one in the finite-dimensional subspace* $\operatorname{span}\{k_{x_1}, ..., k_{x_n}\}$*: there exists* $a \in \mathbb{R}^n$ *such that*

$$\hat{f}_n(x) = \sum_{i=1}^{n} a_i k(x_i, x). \tag{3.5}$$

*If, moreover, $\Omega$ is strictly increasing, every minimizer has this form.*

In the remainder of the chapter, we further assume that $\Omega$ is convex, in order to ensure that (3.3) is a convex optimization problem.

*Proof.* Let $\mathcal{S} \overset{\text{def}}{=} \operatorname{span}\{k_{x_1}, ..., k_{x_n}\}$ and decompose any $f \in \mathcal{H}$ as $f = f_{\mathcal{S}} + f_{\perp}$ with $f_{\mathcal{S}} \in \mathcal{S}$ and $f_{\perp} \in \mathcal{S}^{\perp}$. By the reproducing property, for each $i$,

$$f(x_i) = \langle f, k_{x_i} \rangle_{\mathcal{H}} = \langle f_{\mathcal{S}}, k_{x_i} \rangle_{\mathcal{H}} + \langle f_{\perp}, k_{x_i} \rangle_{\mathcal{H}} = f_{\mathcal{S}}(x_i), \tag{3.6}$$

since $f_{\perp} \perp k_{x_i}$. Hence the empirical term $\hat{L}_n(f)$ depends only on $f_{\mathcal{S}}$. Moreover, by orthogonality,

$$\|f\|_{\mathcal{H}}^2 = \|f_{\mathcal{S}}\|_{\mathcal{H}}^2 + \|f_{\perp}\|_{\mathcal{H}}^2 \geqslant \|f_{\mathcal{S}}\|_{\mathcal{H}}^2, \tag{3.7}$$

so $\|f\|_{\mathcal{H}} \geqslant \|f_{\mathcal{S}}\|_{\mathcal{H}}$, and because $\Omega$ is nondecreasing, we have $\Omega\big(\|f\|_{\mathcal{H}}\big) \geqslant \Omega\big(\|f_{\mathcal{S}}\|_{\mathcal{H}}\big)$. Therefore,

$$\hat{L}_n(f) + \lambda\Omega\big(\|f\|_{\mathcal{H}}\big) \geqslant \hat{L}_n(f_{\mathcal{S}}) + \lambda\Omega\big(\|f_{\mathcal{S}}\|_{\mathcal{H}}\big), \tag{3.8}$$

Replacing $f$ by $f_{\mathcal{S}}$ thus never increases the objective, so a minimizer can be taken in $\mathcal{S}$, which yields (3.5). If $\Omega$ is strictly increasing, then $f_{\perp} \neq 0$ gives $\|f\|_{\mathcal{H}} > \|f_{\mathcal{S}}\|_{\mathcal{H}}$, hence $\Omega\big(\|f\|_{\mathcal{H}}\big) > \Omega\big(\|f_{\mathcal{S}}\|_{\mathcal{H}}\big)$; no minimizer can then have $f_{\perp} \neq 0$, so every minimizer lies in $\mathcal{S}$. □

We let $K \in \mathbb{R}^{n \times n}$ denote the kernel matrix $K_{i,j} \overset{\text{def}}{=} k\big(x_i, x_j\big)$.

**Hinge loss and a first kernel SVM formulation.** For a score function $f : \mathcal{X} \to \mathbb{R}$, we write the classifier as $x \mapsto \operatorname{sign}(f(x))$. It is often useful to include an intercept term, since this extends the hypothesis class from purely linear decision functions in feature space to affine ones, thereby increasing the expressiveness of the model.

A simple way to include an intercept is to augment the feature map with a constant 1. We omit it from the notation for simplicity.

Support vector machines are large-margin models: they favor classifiers for which the signed scores $y_i f(x_i)$ are larger than a prescribed margin (rather than just positive). A standard large-margin loss is the hinge loss

$$\ell_{\mathtt{hinge}}(y_i, f(x_i)) \stackrel{\mathrm{def}}{=} \max(0, 1 - y_i f(x_i)). \tag{3.9}$$

Substituting $\ell_{\mathtt{hinge}}$ into (3.3) and using Theorem 3.1, we obtain the kernel SVM problem with a general regularizer:

$$\hat{a}_n \in \operatorname*{arg\,min}_{a \in \mathbb{R}^n} \frac{1}{n} \sum_{i=1}^{n} \max\bigl(0, 1 - y_i (Ka)_i\bigr) + \lambda \Omega\Bigl(\sqrt{a^\top K a}\Bigr). \tag{3.10}$$

This is the *primal problem in the representer coefficients*.

**Soft-margin SVM: equivalent formulations.** From here on, unless explicitly stated otherwise, we adopt the squared-norm regularizer $\Omega(t) = \frac{1}{2}t^2$, equivalently $\Omega(f) = \frac{1}{2}\|f\|_{\mathcal{H}}^2$. This yields the classical *soft-margin SVM*. Equivalent formulations often expose different algorithmic structures[3], and, before turning to the `snacks` algorithm, it is worth collecting several primal and dual views of the kernel SVM problem. Indeed, the representer theorem lets us parameterize the solution either by the function $w \in \mathcal{H}$ itself (the *functional* view) or by the coefficient vector $a \in \mathbb{R}^n$ of its representer expansion (the *coefficient-space* view). We describe the primal and dual formulations in these two views, and we show that they are equivalent.

**Theorem 3.2.** (Equivalent formulations of the soft-margin kernel SVM) *Let $\Phi : \mathcal{X} \to \mathcal{H}$ be a feature map such that $k(x, x') = \langle \Phi(x), \Phi(x') \rangle_{\mathcal{H}}$. The following optimization problems are equivalent in the sense that they admit the same optimal value and that optimal solutions can be mapped into each other as described in the proof.*

*1.* Coefficient-space primal (unconstrained):

$$\min_{a \in \mathbb{R}^n} \frac{1}{n} \sum_{i=1}^{n} \max\bigl(0, 1 - y_i (Ka)_i\bigr) + \frac{\lambda}{2} a^\top K a. \tag{3.11}$$

*2.* Coefficient-space soft-margin primal (slack variables):

[3] a recurring theme of this thesis

$$\begin{aligned} \min_{a\in\mathbb{R}^n,\,\xi\in\mathbb{R}^n} \quad & \frac{1}{n}\sum_{i=1}^{n}\xi_i + \frac{\lambda}{2}a^\top K a \\ \text{s.t.} \quad & y_i(Ka)_i \geqslant 1-\xi_i, \quad i=1,\dots,n, \\ & \xi_i \geqslant 0, \quad i=1,\dots,n. \end{aligned} \tag{3.12}$$

*3.* RKHS primal (unconstrained):

$$\hat{w} \in \arg\min_{w\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^{n}\max\bigl(0, 1-y_i\langle w, \Phi(x_i)\rangle_{\mathcal{H}}\bigr) + \frac{\lambda}{2}\|w\|_{\mathcal{H}}^2. \tag{3.13}$$

*4.* RKHS soft-margin primal (with slack variables):

$$\begin{aligned} \min_{w\in\mathcal{H},\,\xi\in\mathbb{R}^n} \quad & \frac{1}{n}\sum_{i=1}^{n}\xi_i + \frac{\lambda}{2}\|w\|_{\mathcal{H}}^2 \\ \text{s.t.} \quad & y_i\langle w, \Phi(x_i)\rangle_{\mathcal{H}} \geqslant 1-\xi_i, \quad i=1,\dots,n, \\ & \xi_i \geqslant 0, \quad i=1,\dots,n. \end{aligned} \tag{3.14}$$

*5.* Dual formulation:

$$\begin{aligned} \max_{c\in\mathbb{R}^n} \quad & \lambda c^\top y - \frac{\lambda}{2}c^\top K c \\ \text{subject to} \quad & 0 \leqslant y_i c_i \leqslant \frac{1}{\lambda n}, \quad i=1,\dots,n, \end{aligned} \tag{3.15}$$

*where* $y = (y_1,\dots,y_n)^\top$.

*Proof. Step 1: unconstrained vs. slack-variable formulations.* For any scalar $s\in\mathbb{R}$,

$$\max(0, 1-s) = \min_{\xi\geqslant 0}\xi \quad \text{s.t.} \quad s \geqslant 1-\xi. \tag{3.16}$$

Applying this pointwise with $s = y_i(Ka)_i$ yields the equivalence between (3.11) and (3.12). The same argument with $s = y_i\langle w, \Phi(x_i)\rangle_{\mathcal{H}}$ yields the equivalence between (3.13) and (3.14). In particular, the optimal slack equals $\xi_i = \max\bigl(0, 1-y_i(Ka)_i\bigr)$, the hinge loss of example $i$: it measures how far the example falls short of the target margin, and is zero as soon as $y_i(Ka)_i \geqslant 1$.

*Step 2: coefficient-space vs. RKHS formulations.* For any $a\in\mathbb{R}^n$, define

$$w(a) \stackrel{\text{def}}{=} \sum_{j=1}^{n} a_j\Phi\bigl(x_j\bigr) \in \mathcal{H}. \tag{3.17}$$

Then, for each $i$,

$$\langle w(a), \Phi(x_i)\rangle_{\mathcal{H}} = \sum_{j=1}^{n} a_j \big\langle \Phi(x_j), \Phi(x_i)\big\rangle_{\mathcal{H}} = \sum_{j=1}^{n} a_j k(x_j, x_i) = (Ka)_i, \tag{3.18}$$

and

$$\|w(a)\|_{\mathcal{H}}^2 = \sum_{j=1}^{n}\sum_{\ell=1}^{n} a_j a_\ell \big\langle \Phi(x_j), \Phi(x_\ell)\big\rangle_{\mathcal{H}} = a^\top K a. \tag{3.19}$$

Substituting into (3.13) shows that (3.11) and (3.13) coincide on the data span, hence are equivalent. The same identification yields the equivalence between (3.12) and (3.14).

*Step 3: dual derivation.* Starting from the constrained RKHS problem (3.14), introduce multipliers $\alpha_i \geqslant 0$ for the margin constraints and $\beta_i \geqslant 0$ for $\xi_i \geqslant 0$. The Lagrangian is

$$\begin{aligned} \mathcal{L}(w, \xi, \alpha, \beta) = \frac{1}{n}\sum_{i=1}^{n} \xi_i + \frac{\lambda}{2}\|w\|_{\mathcal{H}}^2 \\ - \sum_{i=1}^{n} \alpha_i\big(y_i \langle w, \Phi(x_i)\rangle_{\mathcal{H}} - 1 + \xi_i\big) - \sum_{i=1}^{n} \beta_i \xi_i. \end{aligned} \tag{3.20}$$

Minimizing in $\xi$ forces $\alpha_i + \beta_i = \frac{1}{n}$ for all $i$, hence $0 \leqslant \alpha_i \leqslant \frac{1}{n}$. Minimizing in $w$ gives

$$\lambda w - \sum_{i=1}^{n} \alpha_i y_i \Phi(x_i) = 0 \quad\Longrightarrow\quad w = \frac{1}{\lambda}\sum_{i=1}^{n} \alpha_i y_i \Phi(x_i). \tag{3.21}$$

To write the dual compactly, we denote the elementwise (Hadamard) product of vectors $u$ and $v$ by $u \odot v$. Substituting back yields the dual problem

$$\begin{aligned} \max_{\alpha \in \mathbb{R}^n} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2\lambda}(\alpha \odot y)^\top K (\alpha \odot y) \\ \text{s.t.} \quad & 0 \leqslant \alpha_i \leqslant \frac{1}{n}, \quad i = 1, \dots, n. \end{aligned} \tag{3.22}$$

Finally, set $c \overset{\text{def}}{=} \frac{1}{\lambda}(\alpha \odot y)$. Then $0 \leqslant y_i c_i = \frac{1}{\lambda}\alpha_i \leqslant \frac{1}{\lambda n}$. Substituting $\sum_{i=1}^{n} \alpha_i = \lambda c^\top y$ and $(\alpha \odot y)^\top K (\alpha \odot y) = \lambda^2 c^\top K c$ into the $\alpha$-dual gives (3.15). The five problems therefore share the same optimal value. □

**Remark.** The dual derivation above relies on the squared RKHS norm, which yields a quadratic regularizer. For a general convex regularizer $\Omega$, the corresponding dual

formulation involves the Fenchel conjugate of $\Omega$ and is naturally expressed using convex duality (Fenchel & Blackett, 1953).

**Support vectors and sparsity.** Write $f(x_i) = \langle w, \Phi(x_i)\rangle_{\mathcal{H}}$ for the optimal score. Complementary slackness for (3.14), together with $\alpha_i + \beta_i = \frac{1}{n}$, classifies each example by its multiplier:

$$\begin{aligned} y_i f(x_i) > 1 &\implies \alpha_i = 0, \\ 0 < \alpha_i < \frac{1}{n} &\implies y_i f(x_i) = 1, \\ \alpha_i = \frac{1}{n} &\implies y_i f(x_i) \leqslant 1, \\ \alpha_i = 0 &\implies y_i f(x_i) \geqslant 1. \end{aligned} \tag{3.23}$$

A point strictly beyond the margin has $\alpha_i = 0$, but a point on the margin ($y_i f(x_i) = 1$) may have $\alpha_i = 0$ or $\alpha_i > 0$. Only examples with $\alpha_i > 0$ enter the expansion $w = \frac{1}{\lambda}\sum_{i=1}^{n} \alpha_i y_i \Phi(x_i)$ with a nonzero coefficient; these are the *support vectors*, and they alone determine the decision function. When they are few, the model is sparse in the dual representation. This cannot be directly leveraged in the primal formulations, so `snacks` does not use it directly.

**Optimization viewpoint: problem classes and basic optimality tools.** The SVM formulations above fit into familiar classes of convex optimization problems. The coefficient-space hinge formulation (3.10) with a general regularizer $\Omega$ is a finite-dimensional *nonsmooth convex* problem in $a$, written as the sum of a piecewise-linear convex loss and a convex regularization term. In the classical squared-norm case $\Omega(t) = \frac{1}{2}t^2$, the primal objectives (3.11) and (3.13) take the form *(nonsmooth) convex loss + smooth quadratic regularizer*. In particular, (3.13) is $\lambda$-strongly convex in $w$ because of the term $\frac{\lambda}{2}\|w\|_{\mathcal{H}}^2$. The slack-variable formulations (3.12) and (3.14) are convex *quadratic programs* (QP) with linear constraints. Finally, the dual problem (3.15) is a *box-constrained QP*, which is well suited to (block) coordinate descent methods.

We now briefly describe the first-order oracles available for the primal objectives and comment on the dual formulation.

Consider first the coefficient-space primal objective

$$F(a) \overset{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} \max\bigl(0, 1 - y_i(Ka)_i\bigr) + \frac{\lambda}{2} a^\top K a. \tag{3.24}$$

A first-order oracle can return the subgradient

$$g(a) \stackrel{\text{def}}{=} \lambda K a - \frac{1}{n} \sum_{i : y_i (Ka)_i < 1} y_i K_{:,i} \in \partial F(a), \tag{3.25}$$

where $K_{:,i}$ denotes the $i$-th column of $K$. When $y_i (Ka)_i = 1$, one may either include or exclude $i$ in the sum; both choices yield valid elements of $\partial F(a)$.

Similarly, for the RKHS primal objective

$$F_{\mathcal{H}}(w) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} \max\bigl(0, 1 - y_i \langle w, \Phi(x_i) \rangle_{\mathcal{H}}\bigr) + \frac{\lambda}{2} \|w\|_{\mathcal{H}}^2, \tag{3.26}$$

a first-order oracle can return the subgradient

$$g_{\mathcal{H}}(w) \stackrel{\text{def}}{=} \lambda w - \frac{1}{n} \sum_{i : y_i \langle w, \Phi(x_i) \rangle_{\mathcal{H}} < 1} y_i \Phi(x_i) \in \partial F_{\mathcal{H}}(w), \tag{3.27}$$

with the same convention for indices satisfying $y_i \langle w, \Phi(x_i) \rangle_{\mathcal{H}} = 1$.

On the dual side, (3.15) is a concave quadratic maximization over a polytope (equivalently, a convex quadratic program after negating the objective). In practice, coordinate and block-coordinate methods proceed by selecting a small block of variables, updating that block by an exact or approximate minimization of the quadratic objective, and enforcing feasibility with respect to the box constraints throughout.

## 3.2 THE SNACKS ALGORITHM

This section presents the snacks algorithm, our main contribution in this chapter. It combines (i) a Nyström approximation of the kernel matrix, (ii) the explicit finite-dimensional feature map arising from this approximation, and (iii) an accelerated stochastic subgradient (ASSG) method (Xu et al., 2019) applied to the resulting linear SVM objective. ASSG and its variants ((R)ASSG-c, (R)ASSG-r) are due to Xu et al. (2019). The contribution of snacks is to compose them with the Nyström embedding into a single large-scale kernel-SVM solver, run with fixed optimizer settings that need no per-dataset tuning (Section 3.4), together with an implementation that scales to datasets whose full kernel matrix would be far too large to store (Section 3.5).

### 3.2.1 Nyström approximation

In kernel SVMs, the main scaling bottleneck is the storage of (and repeated access to) the kernel matrix $K \in \mathbb{R}^{n \times n}$, which costs $\mathcal{O}(n^2)$ memory. Nyström approximation replaces $K$ by a low-rank surrogate constructed from a small subset of $m \ll$

$n$ sampled columns of $K$ (often called Nyström centers or inducing points in the kernel literature).

Fix an index set $\mathcal{J} \subset \{1, ..., n\}$ with $|\mathcal{J}| = m$, indexing the sampled inputs $(x_i)_{i \in \mathcal{J}}$. We define the associated submatrices as $K_{:,\mathcal{J}} \in \mathbb{R}^{n \times m}$ and $K_{\mathcal{J},\mathcal{J}} \in \mathbb{R}^{m \times m}$, and write $K_{\mathcal{J},:} = K_{:,\mathcal{J}}^\top$ for the corresponding rows, by symmetry of $K$. The Nyström approximation of $K$ is

$$\hat{K} \overset{\text{def}}{=} K_{:,\mathcal{J}} K_{\mathcal{J},\mathcal{J}}^\dagger K_{\mathcal{J},:}, \tag{3.28}$$

where $(\cdot)^\dagger$ denotes the Moore-Penrose pseudoinverse. If $m = n$ and $\mathcal{J} = \{1, ..., n\}$, then $\hat{K} = K$. The regime that is the most interesting is $m \ll n$: there $\hat{K}$ has rank at most $m$, so storing only the pair $\left(K_{:,\mathcal{J}}, K_{\mathcal{J},\mathcal{J}}\right)$ determines $\hat{K}$ and costs $\mathcal{O}\left(nm + m^2\right)$ memory rather than the $\mathcal{O}\left(n^2\right)$ that the full matrix costs.

The approximation reproduces the sampled columns exactly:

$$\hat{K}_{:,\mathcal{J}} = K_{:,\mathcal{J}}. \tag{3.29}$$

We apply (3.29) in matrix form. Write $S \in \{0, 1\}^{n \times m}$ for the selection matrix of $\mathcal{J}$, so that $KS = K_{:,\mathcal{J}}$; (3.29) then gives $\hat{K}S = KS$, which is the form the proofs in Chapter 3 use.

To express the Nyström approximation as the Gram matrix of an explicit finite-dimensional embedding, we diagonalize the submatrix $K_{\mathcal{J},\mathcal{J}}$ via eigenvalue decomposition:

$$K_{\mathcal{J},\mathcal{J}} = U \Sigma U^\top, \quad \Sigma = \mathrm{diag}(\sigma_1, ..., \sigma_m), \quad \sigma_j \geqslant 0, \tag{3.30}$$

and define the pseudoinverse square root

$$K_{\mathcal{J},\mathcal{J}}^{\dagger/2} \overset{\text{def}}{=} U \Sigma^{\dagger/2} U^\top, \quad \left(\Sigma^{\dagger/2}\right)_{jj} = \sigma_j^{-1/2} \mathbb{1}_{\{\sigma_j > 0\}}. \tag{3.31}$$

We can then factorize the approximation as

$$\hat{K} = \left(K_{:,\mathcal{J}} K_{\mathcal{J},\mathcal{J}}^{\dagger/2}\right)\left(K_{:,\mathcal{J}} K_{\mathcal{J},\mathcal{J}}^{\dagger/2}\right)^\top. \tag{3.32}$$

This factorization guarantees that $\hat{K}$ is positive semidefinite, allowing it to safely replace $K$ in kernel learning objectives.

The approximation quality depends on how $\mathcal{J}$ is chosen. While uniform sampling without replacement is a simple baseline, more accurate schemes, such as leverage score sampling, bias the selection toward informative columns, reducing the

required $m$ for a target accuracy; see Rudi et al. (2018) for a computationally tractable approximation of these scores.

**Other low-rank approximations.** Nyström is one of several low-rank approximations that yield an explicit feature matrix $Z \in \mathbb{R}^{n \times m}$ such that $K \approx ZZ^\top$. Pivoted Cholesky factorizations, used for SVM training since Fine & Scheinberg (2001), are Nyström approximations whose index set $J$ is selected greedily, one column at a time. Randomized variants such as randomly pivoted Cholesky (Chen et al., 2025) were recently shown to be more reliable than greedy pivoting and to produce near-optimal approximations while evaluating only $\mathcal{O}(nm)$ entries of $K$. It is also possible to directly approximate the kernel function rather than its Gram matrix. This idea underlies the popular random features technique (Rahimi & Recht, 2007), which builds $Z$ from frequencies sampled from the Fourier transform of a shift-invariant kernel, independently of the data. For an RBF kernel on $\mathbb{R}^d$, Nyström and pivoted Cholesky cost $\mathcal{O}(nm(d+m))$ operations, $\mathcal{O}(nm)$ kernel evaluations followed by $\mathcal{O}(nm^2)$ linear algebra, while random features cost $\mathcal{O}(nmd)$. Hence, for a fixed $m$, random features are cheaper when $m$ exceeds $d$; however, the methods may need different values of $m$ to reach a given accuracy. Finally, since `snacks` only uses $Z$, any of these embeddings can replace Nyström without changing the optimizer.

**Numerical stability.** When $K_{J,J}$ is ill-conditioned, computing $K_{J,J}^{\dagger/2}$ may be unstable. In practice, several strategies can mitigate this:

1. *Truncation*: discarding eigenvalues below $\tau\sigma_{max}$, where $\sigma_{max}$ is the largest eigenvalue and $\tau > 0$, to retain only $r \leqslant m$ components;
2. *Ridge*: replacing $K_{J,J}$ with $K_{J,J} + \mu I_m$ for a small $\mu > 0$ prior to factorization.

Both strategies improve numerical robustness while preserving the computational efficiency of the low-rank approximation (Epperly et al., 2024; Li et al., 2017; Tropp et al., 2017). The kernel matrix $K_{J,J}$ suffers from ill-conditioning as soon as some sampled columns are nearly redundant. This is partly unavoidable, since the spectrum of the Gaussian kernel decays quickly and even *well-spread* sampled columns can give tiny eigenvalues as $m$ grows. Experimental-design criteria such as D-optimality (that is, maximizing $\det K_{J,J}$) help in sampling columns which avoid this redundancy. Truncation achieves a similar effect after the fact, by discarding the redundant directions of uniformly sampled columns. In the remainder we use truncation; the embedding then has a fixed dimension $r \leqslant m$ (with $r = m$ when nothing is discarded), which we adopt from here on.

### 3.2.2 Kernel data embedding

We now reformulate (3.32) into an explicit feature map. Let $U_r \in \mathbb{R}^{m\times r}$ contain the retained eigenvectors, and let $\Sigma_r \in \mathbb{R}^{r\times r}$ contain the retained positive eigenvalues. We define

$$Z \overset{\text{def}}{=} K_{:,J} U_r \Sigma_r^{-1/2} \in \mathbb{R}^{n\times r}, \quad \text{so that} \quad \hat{K}_r = ZZ^\top. \tag{3.33}$$

Here $\hat{K}_r$ is the Nyström approximation we stabilize by truncating the spectrum of $K_{J,J}$ at rank $r$. The factorization in (3.33) is *one-sided*: we right-multiply $U_r$ by $\Sigma_r^{-1/2}$ and apply no factor on the left, so the $m \times r$ matrix $W \overset{\text{def}}{=} U_r\Sigma_r^{-1/2}$ carries the embedding and $Z = K_{:,J}W$. A point $x$ enters through its kernel values on the sampled columns, $k_J(x) \in \mathbb{R}^m$, and embeds as $W^\top k_J(x)$, so $W$ also handles points outside the training set. A symmetric alternative is equally natural. Writing the truncated symmetric inverse square root as $U_r\Sigma_r^{-1/2}U_r^\top$, one may instead take

$$Z_{sym} \overset{\text{def}}{=} K_{:,J} U_r \Sigma_r^{-1/2} U_r^\top = ZU_r^\top. \tag{3.34}$$

The two factors coincide up to an orthogonal rotation on the retained eigenspace, and both yield the same approximation, $Z_{sym}Z_{sym}^\top = ZZ^\top = \hat{K}_r$. We use the one-sided factor $Z$, as it directly gives the $r$-dimensional representation. The symmetric factor does not give this reduction, as the trailing $U_r^\top$ restores the ambient column dimension and leaves $Z_{sym}$ of size $n \times m$. The matrix $Z$ can be viewed as the feature matrix for the training data induced by the $m$ sampled columns. Let $z_i^\top$ denote the $i$-th row of $Z$, then $\left(\hat{K}_r\right)_{i,j} = \langle z_i, z_j\rangle_{\mathbb{R}^r}$.

For any input $x \in \mathcal{X}$, define the vector of kernel evaluations against the sampled columns (in the fixed order of $J$)

$$k_J(x) \overset{\text{def}}{=} \big(k(x_{i_1}, x), \ldots, k(x_{i_m}, x)\big)^\top \in \mathbb{R}^m, \tag{3.35}$$

and the embedding map $\psi : \mathcal{X} \to \mathbb{R}^r$ by

$$\psi(x) \overset{\text{def}}{=} \Sigma_r^{-1/2} U_r^\top k_J(x) = W^\top k_J(x). \tag{3.36}$$

For training points, $z_i = \psi(x_i)$ and $\left(\hat{K}_r\right)_{i,j} = \big\langle \psi(x_i), \psi(x_j)\big\rangle_{\mathbb{R}^r}$ by construction.

**Reduction to linear SVM.** By leveraging the feature map $\psi$, the Nyström-approximated kernel SVM is reformulated as a linear SVM in $\mathbb{R}^r$. Define the linear predictor

$$f_u(x) \overset{\text{def}}{=} \langle u, \psi(x)\rangle, \tag{3.37}$$

and consider the objective

$$\begin{aligned}\min_{u\in\mathbb{R}^r} F(u) &\overset{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} \max(0, 1 - y_i f_u(x_i)) + \frac{\lambda}{2}\|u\|_2^2 \\ &= \frac{1}{n}\sum_{i=1}^{n} \max\Big(0, 1 - y_i z_i^\top u\Big) + \frac{\lambda}{2}\|u\|_2^2.\end{aligned} \tag{3.38}$$

This objective is exactly the kernel SVM with $\hat{K}_r$ in place of $K$. Indeed, applying the coefficient-space primal of Theorem 3.2 to $\hat{K}_r$ gives

$$\min_{a\in\mathbb{R}^n} \frac{1}{n}\sum_{i=1}^{n} \max\Big(0, 1 - y_i\Big(\hat{K}_r a\Big)_i\Big) + \frac{\lambda}{2} a^\top \hat{K}_r a, \tag{3.39}$$

and the substitution $u = Z^\top a$ yields $\left(\hat{K}_r a\right)_i = \left(ZZ^\top a\right)_i = z_i^\top u$ and $a^\top \hat{K}_r a = \left\|Z^\top a\right\|_2^2 = \|u\|_2^2$, so the objective becomes $F(u)$. As $a$ ranges over $\mathbb{R}^n$, $u = Z^\top a$ ranges over the row space of $Z$, which is all of $\mathbb{R}^r$ since $Z$ has rank $r$; the two problems therefore share the same optimal value, with solutions related by $u = Z^\top a$. Equation (3.38) serves as the optimization problem for our proposed algorithm. The resulting classifier is $\hat{y}(x) = \operatorname{sign}(f_u(x))$.

**Remark.** (Offline embedding vs. on-the-fly kernel operations) The definition of $\psi$ in (3.36) highlights a computational tradeoff. One strategy is to keep the Nyström representation implicit and evaluate $k_J(x)$ on the fly during each iteration. This minimizes memory usage but incurs the cost of repeated kernel evaluations. That keeps memory at $\mathcal{O}(m^2)$, but every gradient step then evaluates the kernel $m$ times to rebuild the row it needs. We instead precompute $Z$ once. While this requires $\mathcal{O}(nr)$ memory, it reduces subsequent stochastic gradient updates to efficient $\mathcal{O}(r)$ linear operations.

### 3.2.3 An accelerated stochastic subgradient method

We focus on minimizing the objective $F(u)$ defined in (3.38). This problem is $\lambda$-strongly convex but nonsmooth due to the hinge loss component. `snacks` solves it with the restarted accelerated stochastic subgradient method RASSG-r of Xu et al. (2019).

**Stochastic oracle and bounds.** Sampling $i \sim \mathrm{Unif}\{1, \dots, n\}$ and setting

$$g(u; i) \overset{\text{def}}{=} \lambda u - y_i z_i \mathbb{1}\Big\{y_i z_i^\top u < 1\Big\} \tag{3.40}$$

yields an unbiased subgradient estimate of $F$, computable in $\mathcal{O}(r)$ operations. Set $M \stackrel{\text{def}}{=} \max_{i=1,\dots,n} \|z_i\|_2$. Every subgradient of the loss has norm at most $M$, so $0 \in \partial F(u^*)$ forces $\|\lambda u^*\|_2 \leqslant M$; hence $u^*$ lies in the ball $\mathcal{B}_R \stackrel{\text{def}}{=} \{u \in \mathbb{R}^r : \|u\|_2 \leqslant R\}$ with $R \stackrel{\text{def}}{=} M/\lambda$. On $\mathcal{B}_R$, $\|g(u;i)\|_2 \leqslant \lambda R + M = 2M$.

**From SSG to localized acceleration.** A standard baseline is the projected stochastic subgradient method (SSG), which iterates

$$u_{t+1} = \Pi_{\mathcal{B}_R}(u_t - \eta_t g(u_t; i_t)). \tag{3.41}$$

For $\lambda$-strongly convex objectives with bounded subgradients, standard analyses establish an $\mathcal{O}(1/T)$ convergence rate (up to logarithmic factors, depending on the averaging scheme) using stepsizes $\eta_t \propto 1/(\lambda t)$ (Lacoste-Julien et al., 2012). However, high-probability analyses show that the variance of the stochastic subgradient depends on the distance from the intermediate iterates to the optimal solution (Juditsky & Nesterov, 2014): even when the subgradients are bounded, a large feasible diameter can amplify stochastic fluctuations. ASSG (Xu et al., 2019) therefore confines the iterates to progressively smaller neighborhoods. Once an iterate lies close to $u^*$, restricting SSG to a small ball around that iterate bounds how far the stochastic noise can move it, and strong convexity keeps $u^*$ inside the ball as its radius decreases. Note that in the original paper, Xu et al. (2019) use this stagewise localization to *accelerate* the stochastic subgradient method, in the more general setting of Hölderian error bounds, which still guarantee that a ball contains the solution.

There are two ways to do localization. Both proceed in stages: a stage runs an inner loop, averages its iterates into the stage output $u^{(k)}$, and warm-starts the next stage from it, so the center is $c_k \stackrel{\text{def}}{=} u^{(k-1)}$. The constrained variant (ASSG-c) projects onto the intersection $\mathcal{B}_R \cap \mathcal{B}(u^{(k-1)}, D_k)$, where $\mathcal{B}(c, D) \stackrel{\text{def}}{=} \{u : \|u - c\|_2 \leqslant D\}$. This projection requires a dedicated subroutine whose cost scales as $\mathcal{O}(m)$, as much as the stochastic oracle (3.40) itself, and the radii $\{D_k\}$ must be chosen. The regularized variant (ASSG-r) regularizes toward $c_k$ instead: it runs stochastic subgradient descent on the strongly convex stage objective $F(v) + \frac{1}{2\beta_k}\|v - c_k\|_2^2$, with stage scale $\beta_k > 0$. Writing $F = F_0 + \frac{\lambda}{2}\|\cdot\|_2^2$, where $F_0$ is the average hinge loss, the two quadratic terms merge:

$$F(v) + \frac{1}{2\beta_k}\|v - c_k\|_2^2 = F_0(v) + \frac{1}{2\beta_k'}\|v - c_k'\|_2^2 + \text{const}, \tag{3.42}$$

with $\beta_k' \stackrel{\text{def}}{=} \beta_k/(1 + \lambda\beta_k)$ and $c_k' \stackrel{\text{def}}{=} c_k/(1 + \lambda\beta_k)$. A stage is therefore an ASSG-r stage on $F_0$ alone, with center $c_k'$ and parameter $\beta_k'$, and its stochastic subgradient

$$h(v;i) \stackrel{\text{def}}{=} -y_i z_i \mathbb{1}\left\{y_i z_i^\top v < 1\right\} \tag{3.43}$$

has norm at most $M$ on all of $\mathbb{R}^r$. No projection is used: by Lemma 4 of Xu et al. (2019) the regularizer alone keeps each stage's iterates within $2\beta'_k M$ of the center $c'_k$. The ball $\mathcal{B}_R$ enters only in the analysis, which uses it to locate $u^*$. Each iteration is then a single affine combination of the current iterate, the center $c'_k$, and the stochastic subgradient $h$ (Algorithm 3.1). Hence, `snacks` uses the regularized variant ASSG-r.

**Restarts: RASSG-r.** RASSG-r restarts the regularized method. Restart $s$ starts from the previous restart's last iterate, runs $J$ stages of length $T_s = \lfloor T_0 \gamma^{s-1} \rceil$, sets $\beta = c_\beta/(\lambda\sqrt{T_s})$, and divides $\beta$ by $\rho_\beta$ after each stage. A single, monotonically decreasing $\beta$ slows the method; the restart raises $\beta$ again, avoiding this without losing localization.

**Algorithm 3.1.** (RASSG-r: Restarted Accelerated Stochastic Subgradient (regularized form))

**Inputs:** restarts $S$, stages per restart $J$, base horizon $T_0$, growth $\gamma$, scale $c_\beta$, decay $\rho_\beta$
**Initialization:** $u \leftarrow 0$
**for** $s = 1, \ldots, S$:
  $T_s \leftarrow \lfloor T_0 \gamma^{s-1} \rceil, \quad \beta \leftarrow c_\beta/(\lambda\sqrt{T_s})$
  **for** $k = 1, \ldots, J$:
    $c' \leftarrow u/(1+\lambda\beta), \quad \beta' \leftarrow \beta/(1+\lambda\beta), \quad v_1 \leftarrow c'$
    **for** $\tau = 1, \ldots, T_s - 1$:
      Sample $i \sim \mathrm{Unif}\{1, \ldots, n\}$
      $v_{\tau+1} \leftarrow \left(1 - \frac{2}{\tau}\right) v_\tau + \frac{2}{\tau} c' - \frac{2\beta'}{\tau} h(v_\tau; i)$
    $u \leftarrow \mathrm{average}(v_1, \ldots, v_{T_s})$
    $\beta \leftarrow \beta/\rho_\beta$
**Output:** best validation iterate among the stage outputs (**or** the last iterate)

### 3.2.4 Putting the pieces together

We now have the embedded objective (3.38), the stochastic subgradient (3.43) of its loss, and the optimizer Algorithm 3.1. Program 3.1 combines them into a reference implementation of `snacks`.

**Program 3.1.** (snacks) Minimal reference implementation of `snacks`.

```
def snacks(X, y, kernel, lam, m, assg_params, tau=1e-4,
rng=None):
    # Sample m Nystrom centers and form the kernel blocks
    rng = np.random.default_rng(rng)
    I = rng.choice(len(y), size=m, replace=False)
    K_XI, K_II = kernel(X, X[I]), kernel(X[I], X[I])
    # Eigendecompose K_II and keep the eigenvalues above
tau * sigma_max
    sigma, U = np.linalg.eigh(K_II)
    keep = sigma > tau * sigma.max()
    sigma, U = sigma[keep], U[:, keep]
    # Nyström features
    Z = (K_XI @ U) / np.sqrt(sigma)  # Shape (n, r)
    # Stochastic subgradient of the hinge loss; rassg_r
handles lam
    def h(u, i):
        zi = Z[i]
        if y[i] * (zi @ u) < 1.0:
            return -y[i] * zi
        return np.zeros_like(u)
    u0 = np.zeros(Z.shape[1])
    u = rassg_r(h, lam=lam, u0=u0, **assg_params)
    return u, I, U, sigma
```

## 3.3 Computational cost of **snacks**

We now analyze the computational complexity of `snacks`, focusing on dominant arithmetic operations and kernel evaluations. We count one flop per floating-point addition, subtraction, or multiplication, ignoring comparisons and memory access costs. Kernel evaluations are accounted for separately: let $\kappa_k$ denote the flop cost of evaluating $k(x, x')$.

Throughout this section, we assume no rank truncation (i.e., $r = m$), so $Z \in \mathbb{R}^{n \times m}$.

**Theorem 3.3.** (Dominant computational cost of snacks) *Let $m$ be the size of the sampled column subset $\mathcal{I}$ and let $\kappa_k$ be the flop cost of one kernel evaluation. The dominant costs of* snacks *admit the following $\mathcal{O}(\cdot)$ bounds:*

***(i) One-time preprocessing.*** *Forming the Nyström features $Z$ and computing $M = \max_i \|z_i\|_2$ costs*

$$\mathrm{Cost}_{\mathrm{prep}} = \mathcal{O}\big(\kappa_k(nm + m^2) + m^3 + nm^2\big). \tag{3.44}$$

***Remark.*** *Since $K_{\mathcal{I},\mathcal{I}}$ is symmetric, it can be formed by evaluating the kernel only on the upper triangle and mirroring. This reduces the number of kernel evaluations from $m^2$ to $\frac{m(m+1)}{2}$, affecting only the constant factors.*

***(ii) Optimization cost for the restarted schedule.*** *RASSG-r runs $S$ restarts, each comprising $J$ stages, with restart $s$ using the inner horizon $T_s = \lfloor T_0 \gamma^{s-1} \rceil$. Since each inner update requires $\mathcal{O}(m)$ operations, the total optimization cost is*

$$\mathrm{Cost}_{\mathrm{opt}} = \mathcal{O}\left( m \sum_{s=1}^{S} JT_s \right) = \mathcal{O}\left( mJT_0 \frac{\gamma^S - 1}{\gamma - 1} \right). \tag{3.45}$$

***(iii) Optimization cost to reach accuracy.*** *Fix a target precision $\varepsilon \in (0, 1]$ and confidence $1 - \delta$, with $\delta \in (0, 1)$. Each stage of RASSG-r is an ASSG-r stage on the hinge loss $F_0$ alone, the ridge term being absorbed into the stage center and parameter, so the analysis only needs a bound on the stochastic subgradient (3.43) of $F_0$. It satisfies $\|h(v; i)\|_2 \leqslant M$ on all of $\mathbb{R}^r$, hence $G \overset{\mathrm{def}}{=} M$. To guarantee an $\varepsilon$-suboptimal solution with probability at least $1 - \delta$, the analysis in Xu et al. (2019) requires setting:*

$$J = \mathcal{O}(\log(1/\varepsilon)), \quad T = \tilde{\mathcal{O}}\left( \frac{G^2}{\lambda \varepsilon} \right), \tag{3.46}$$

*where $T$ is the per-stage horizon. Substituting these values into the single-run budget $\mathcal{O}(mJT)$ gives the total complexity*

$$\mathrm{Cost}_{\mathrm{acc}} = \tilde{\mathcal{O}}\left( m \frac{G^2}{\lambda \varepsilon} \log(1/\varepsilon) \right), \tag{3.47}$$

*where $\tilde{\mathcal{O}}$ suppresses logarithmic factors in $\varepsilon^{-1}$ and $\delta^{-1}$.*

***(iv) Prediction cost.*** *For a test point $x \in \mathcal{X}$, computing the Nyström feature map $\psi(x)$ requires*

$$\mathrm{Cost}_{\psi}(x) = \mathcal{O}\big(\kappa_k m + m^2\big) \quad \text{flops}, \tag{3.48}$$

*and computing the score* $f_u(x) = u^\top \psi(x)$ *costs an additional* $\mathcal{O}(m)$.

***Remark.*** *If only the prediction score is required (rather than the explicit feature map), one can precompute the transformed weight vector*

$$w \stackrel{\text{def}}{=} U\Sigma^{\dagger/2}u \in \mathbb{R}^m \tag{3.49}$$

*once after training. This operation costs* $\mathcal{O}(m^2)$. *Subsequent predictions are then computed via*

$$f_u(x) = w^\top k_J(x), \tag{3.50}$$

*reducing the per-point cost to* $\mathcal{O}(\kappa_k m + m)$. *If* $x = x_i$ *is a training point for which* $z_i$ *is cached, prediction reduces to a single dot product costing* $\mathcal{O}(m)$.

*Proof.* We first analyze the dominant complexity terms for preprocessing then for optimization, and finally for prediction.

**Preprocessing (Nyström).** The construction of the Nyström approximation involves three steps:

1. *Kernel evaluations*: Forming the cross-kernel block $K_{:,J}$ requires $\mathcal{O}(\kappa_k nm)$ operations. Evaluating the principal block $K_{J,J}$ costs a further $\mathcal{O}(\kappa_k m^2)$ operations, dominated by the previous term since $m \leqslant n$.
2. *Factorization*: Computing the eigendecomposition $K_{J,J} = U\Sigma U^\top$ costs $\mathcal{O}(m^3)$.
3. *Features*: Computing $Z = K_{:,J}U\Sigma^{\dagger/2}$ costs $\mathcal{O}(nm^2)$ for the matrix multiplication, and $\mathcal{O}(nm)$ for the subsequent scaling by $\Sigma^{\dagger/2}$.

Finally, computing the radius $M = \max_i \|z_i\|_2$ costs $\mathcal{O}(nm)$. Summing these terms yields the bound in (i).

**Optimization (RASSG-r).** Each inner iteration has two steps:

1. a stochastic subgradient evaluation costing $\mathcal{O}(m)$;
2. a proximal-centering step (a convex combination of the current iterate, the stage center, and the subgradient), also costing $\mathcal{O}(m)$.

RASSG-r runs S restarts of J stages each, with restart s using $T_s = \lfloor T_0 \gamma^{s-1} \rfloor$ inner iterations, so the total cost is

$$\text{Cost}_{\text{opt}} = \mathcal{O}\left( m \sum_{s=1}^{S} JT_s \right) = \mathcal{O}\big(mJT_0(\gamma^S - 1)/(\gamma - 1)\big) \tag{3.51}$$

.

**Complexity to reach accuracy.** Consider the embedded SVM objective (in Nyström space)

$$F(u) = \frac{\lambda}{2}\|u\|_2^2 + \frac{1}{n}\sum_{i=1}^{n} \max\Big(0, 1 - y_i z_i^\top u\Big), \quad u \in \mathbb{R}^m. \tag{3.52}$$

The average hinge loss $F_0$ is $M$-Lipschitz on $\mathbb{R}^m$, and $F$ is $\lambda$-strongly convex on $\mathbb{R}^m$. For a single accelerated run, the analysis of Xu et al. (2019) sets $J = \mathcal{O}(\log(1/\varepsilon))$ stages and a per-stage horizon $T = \tilde{\mathcal{O}}\big(M^2/(\lambda\varepsilon)\big)$ to reach an $\varepsilon$-suboptimal solution with high probability. Combined with the per-iteration cost from (ii), this gives the bound in (iii).

**Prediction.** Given an input $x$, computing the kernel vector $k_J(x)$ costs $\mathcal{O}(\kappa_k m)$. Forming $\psi(x) = \Sigma^{\dagger/2} U^\top k_J(x)$ then costs $\mathcal{O}(m^2)$. The score $u^\top \psi(x)$ adds $\mathcal{O}(m)$, which establishes (iv). If the score is all that is required, form $w = U\Sigma^{\dagger/2}u$ once; each prediction is then $w^\top k_J(x)$, at $\mathcal{O}(\kappa_k m + m)$. □

**Remark.** *Comparison with dual methods.* Dual coordinate descent (DCD) (Hsieh et al., 2008) and stochastic dual coordinate ascent (SDCA) (Shalev-Shwartz & Zhang, 2013) both solve the dual of (3.38) over $\alpha \in [0, 1/n]^n$ and maintain $w = \frac{1}{\lambda}\sum_{i=1}^{n} \alpha_i y_i z_i$. Updating $\alpha_i$ only requires $z_i^\top w$ and a closed-form step, after which $w$ changes by a multiple of $z_i$; each update therefore costs $\mathcal{O}(m)$, just as primal stochastic subgradient steps. To reach $\varepsilon$-accuracy, DCD requires $\mathcal{O}(n \log(1/\varepsilon))$ updates and SDCA requires $\tilde{\mathcal{O}}\big(n + M^2/(\lambda\varepsilon)\big)$ updates, against $\tilde{\mathcal{O}}\big(M^2/(\lambda\varepsilon)\big)$ for `snacks` in (iii).

## 3.4 Practical implementation

`snacks` is available as a `scikit-learn` (Pedregosa et al., 2011) estimator in the `snacks` Python package. We describe the computational choices behind the implementation; the codebase and documentation give further detail.

### 3.4.1 Architectural design

Training proceeds in two steps: build the Nyström embedding, then minimize the regularized hinge loss over it.

**Nyström embedding.** We sample $m$ columns $X_J$ and form the spectral decomposition of $K_{J,J} + \mu I_m$, with a small ridge $\mu = 10^{-6}$. We truncate that decomposition, discarding the eigenvalues below $\tau\sigma_{\max}$, where $\sigma_{\max}$ is the largest one (default $\tau = 10^{-4}$), and inverting the square root of only those that remain. The embedding is then $\psi(x) = \mathrm{diag}\big(\sigma^{-1/2}\big) U^\top k_J(x)$. The construction touches only the $n \times m$

cross-kernel, never the full $n \times n$ matrix, so it stays feasible when $K$ exceeds memory. The fitted predictor depends on $\mathcal{O}(mr + md)$ parameters, independently of $n$.

**Solver backend.** RASSG-r is implemented as a single compiled Cython routine. Because the whole optimization loop runs in compiled code, no Python interpreter overhead is paid per iteration, and memory access stays contiguous. Stage outputs are obtained by averaging inner iterates from the end of the stage.

### 3.4.2 Optimization schedule and heuristics

**Hyperparameter configuration.** Table 3.1 lists the default parameters. Every parameter in the table is fixed except the Nyström rank $m$, which we select per dataset (Section 3.5). For the optimizer, the stepsize parameter $\beta$ follows from the regularization parameter $\lambda$ and the inner horizon $T_s$, the restart schedule is fixed in advance.

Table 3.1: Hyperparameter configuration.

| | Parameter | Default | Rationale |
|---|---|---|---|
| Nyström | $m$ | per dataset | Approximation rank vs. cost |
| | $\mu$ | $10^{-6}$ | Ridge shift |
| | $\tau$ | $10^{-4}$ | Spectral truncation (relative to $\sigma_{max}$) |
| RASSG-r | $\lambda$ | $10^{-4}$ | Regularization strength |
| | Restarts S | 4 | Outer restarts |
| | Stages/restart J | 10 | Stages per restart |
| | Initial horizon $T_0$ | 512 | Inner iterations (restart 1) |
| | Growth $\gamma$ | 2 | Horizon growth per restart |
| | Scale $c_\beta$ | 10 | Sets $\beta = c_\beta/(\lambda\sqrt{T_s})$ |
| | Decay $\rho_\beta$ | 1.35 | Per-stage $\beta$ decay |

Although the number of iterations $T_0 = 512$ can appear small, we note that (1) it is doubled with each restart and (2) it appeared sufficient to reach the best validation accuracy on all datasets in our experiments.

## 3.5 Large-scale numerical experiments

We now turn to the empirical behavior of `snacks`. The preceding sections show that replacing the full kernel matrix by a Nyström embedding changes the storage and arithmetic costs from quantities involving $n^2$ to quantities involving $nm$ and $m^2$. The purpose of this section is to verify that this computational gain does not destroy the predictive performance of the resulting SVM, and to isolate the contribution of the accelerated stochastic optimizer from the Nyström approximation itself.

The experiments are organized around four questions. First, for a fixed Nyström embedding, how does `snacks` compare with standard linear SVM solvers applied to the same embedded data? Second, how does the runtime scale with the number of training examples? Third, how sensitive is the method to the Nyström rank $m$? Finally, does the restarted, accelerated RASSG-r schedule improve over a non-accelerated stochastic subgradient baseline on the same objective?

### 3.5.1 Protocol

**Datasets and preprocessing.** We use binary classification datasets from the LIBSVM collection (Chang & Lin, 2011), listed in Table 3.2, with labels in $\{-1, +1\}$. Each is split into training, validation, and test sets, keeping the official test set where one exists. Dense features are standardized on the training set, and the non-binary datasets (MNIST, MSD) are binarized beforehand.

Table 3.2: Datasets used in the experiments, ordered by training size. The last column is the memory needed to store the dense $n_{\mathrm{train}} \times n_{\mathrm{train}}$ kernel matrix in double precision ($8n_{\mathrm{train}}^2/2^{30}$ GiB, to three significant figures), which the Nyström embedding avoids.

| Dataset | $n_{\mathrm{train}}$ | $n_{\mathrm{test}}$ | d | $K_{\mathrm{train}}$ GiB |
|---|---|---|---|---|
| splice | 800 | 2 175 | 60 | 0.00477 |
| mnist | 36 000 | 12 000 | 779 | 9.66 |
| w8a | 39 799 | 14 951 | 300 | 11.8 |
| ijcnn1 | 39 992 | 91 701 | 22 | 11.9 |
| epsilon | 320 000 | 100 000 | 2 000 | 763 |
| covtype | 348 607 | 116 203 | 54 | 905 |
| MSD | 370 972 | 51 630 | 90 | 1 030 |
| SUSY | 3 600 000 | 500 000 | 18 | 96 600 |
| HIGGS | 8 400 000 | 500 000 | 28 | 526 000 |

**Methods.** We compare `snacks` against three baselines. *LibLinear* solves the dual linear SVM by coordinate descent (Fan et al., 2008), and *Pegasos* the embedded primal objective by stochastic subgradient steps (Shalev-Shwartz et al., 2011). These two receive exactly the inputs `snacks` does: the same Nyström columns, RBF bandwidth, regularization parameter, and embedded train, validation, and test matrices. Their comparison therefore isolates the optimizer from the kernel approximation. The third, *sklearn-Nyström*[4], instead builds its own embedding and fits a linear SVM on it (Pedregosa et al., 2011). It is included as the standard tool used to train a kernel SVM at scale.

**Hyperparameter selection.** All experiments use the RBF kernel, whose bandwidth is set by the median-bandwidth heuristic. The regularization parameter is selected from

$$\lambda \in \{3 \cdot 10^{-5}, 10^{-4}, 3 \cdot 10^{-4}, 10^{-3}, 3 \cdot 10^{-3}, 10^{-2}, 3 \cdot 10^{-2}, 10^{-1}\}, \quad (3.53)$$

and the number of sampled Nyström columns by grid search over

$$m \in \{100, 200, 500, 1000, 2000, 5000\}, \quad (3.54)$$

with ranks larger than the training set removed. On the largest datasets we do not sweep the grid: accuracy keeps improving with the rank, so we simply take the largest $m$ that fits in memory for all methods. For each method we keep the configuration with the highest mean validation accuracy across seeds, and evaluate it on the held-out test set.

**Metrics and timing.** We compare methods by the test accuracy they reach. We report wall-clock training time as the sum of the Nyström embedding time and the solver time. Table entries are averaged over five independent runs. All experiments were run on a cluster, each run using 8 cores of a node with a 32-core AMD EPYC 7551P processor and 256 GB of RAM.

### 3.5.2 Speed and accuracy across methods

We first compare the four methods in terms of training time and test error on the datasets of Table 3.2. LibLinear, Pegasos, and `snacks` receive the same sampled columns, bandwidth, regularization parameter, and embedded data, so any difference between them comes from the optimizer. sklearn-Nyström, the standard tool

[4]The original `snacks` paper (Tanji et al., 2023) used *ThunderSVM* (Wen et al., 2018) as a baseline. That code is no longer maintained and does not build as is; once patched, its CPU results were not competitive with the simpler and more widely used sklearn-Nyström, which we report instead.

for training a kernel SVM at scale, instead builds its own embedding. Table 3.3 reports the total training time and test error of the four methods.

Table 3.3: Comparison across data sets and methods, one row per data set and one column per method. Data sets are ordered by training size $n$. The upper panel reports total training time in seconds, the lower panel the percentage of misclassified test points; each entry is the mean $\pm$ standard deviation over runs, and smaller is better in both. The Nyström rank $m$ is shared by all four methods on a given data set. Bold marks the best mean in each row.

**Training time (s), smaller is better**

| Data set | $m$ | LibLinear | Pegasos | sklearn-Nyström | `snacks` |
|---|---|---|---|---|---|
| splice | 500 | $0.3 \pm 0.1$ | $\mathbf{0.1 \pm 0.0}$ | $0.5 \pm 0.1$ | $\mathbf{0.1 \pm 0.0}$ |
| mnist | 5 000 | $28.7 \pm 1.9$ | $26.7 \pm 1.8$ | $56.7 \pm 3.4$ | $\mathbf{25.9 \pm 1.8}$ |
| w8a | 5 000 | $36.3 \pm 3.7$ | $27.4 \pm 2.6$ | $133.7 \pm 51.0$ | $\mathbf{26.8 \pm 2.6}$ |
| ijcnn1 | 5 000 | $19.9 \pm 0.5$ | $19.1 \pm 0.6$ | $51.2 \pm 3.8$ | $\mathbf{19.0 \pm 0.6}$ |
| epsilon | 2 000 | $87.3 \pm 15.6$ | $76.7 \pm 14.6$ | $114.6 \pm 14.1$ | $\mathbf{63.8 \pm 14.1}$ |
| covtype | 5 000 | $56.6 \pm 3.7$ | $54.8 \pm 3.9$ | $219.5 \pm 8.0$ | $\mathbf{53.4 \pm 3.7}$ |
| MSD | 5 000 | $74.4 \pm 1.8$ | $69.6 \pm 0.4$ | $221.5 \pm 16.0$ | $\mathbf{60.9 \pm 0.2}$ |
| SUSY | 500 | $87.1 \pm 10.3$ | $66.3 \pm 5.8$ | $133.4 \pm 9.0$ | $\mathbf{33.3 \pm 2.0}$ |
| HIGGS | 200 | $149.9 \pm 1.1$ | $169.8 \pm 23.9$ | $144.8 \pm 14.3$ | $\mathbf{32.6 \pm 3.0}$ |

**Test error (%), smaller is better**

| Data set | $m$ | LibLinear | Pegasos | sklearn-Nyström | `snacks` |
|---|---|---|---|---|---|
| splice | 500 | $\mathbf{11.9 \pm 0.3}$ | $12.3 \pm 0.7$ | $12.0 \pm 0.6$ | $12.0 \pm 0.5$ |
| mnist | 5 000 | $3.8 \pm 0.1$ | $3.9 \pm 0.1$ | $\mathbf{3.6 \pm 0.0}$ | $4.1 \pm 0.0$ |
| w8a | 5 000 | $2.2 \pm 0.0$ | $2.2 \pm 0.0$ | $2.2 \pm 0.0$ | $\mathbf{2.0 \pm 0.1}$ |
| ijcnn1 | 5 000 | $6.0 \pm 0.0$ | $5.9 \pm 0.1$ | $5.7 \pm 0.0$ | $\mathbf{5.2 \pm 0.3}$ |
| epsilon | 2 000 | $14.3 \pm 0.1$ | $14.4 \pm 0.1$ | $\mathbf{14.1 \pm 0.0}$ | $14.9 \pm 0.1$ |
| covtype | 5 000 | $25.6 \pm 0.1$ | $25.6 \pm 0.2$ | $\mathbf{25.4 \pm 0.2}$ | $\mathbf{25.4 \pm 0.3}$ |
| MSD | 5 000 | $33.6 \pm 0.1$ | $33.6 \pm 0.2$ | $33.2 \pm 0.0$ | $\mathbf{32.9 \pm 0.3}$ |
| SUSY | 500 | $20.4 \pm 0.0$ | $20.4 \pm 0.0$ | $\mathbf{20.3 \pm 0.0}$ | $\mathbf{20.3 \pm 0.0}$ |
| HIGGS | 200 | $36.9 \pm 0.1$ | $\mathbf{36.4 \pm 0.1}$ | $36.8 \pm 0.1$ | $37.3 \pm 0.1$ |

To make the timing source explicit, Table 3.4 separates the shared Nyström embedding time from the remaining solver-side training time for the three methods that expose this split.

Table 3.4: Decomposition of training time, in seconds, for the methods using the shared explicit Nyström features. Embedding covers computing the Nyström approximation from the sampled columns and embedding the training set; the remaining columns report total training time minus this embedding cost. For LibLinear and Pegasos this residual is the linear SVM fit, while for `snacks` it includes the optimizer and validation-stage selection overhead. The sklearn-Nyström baseline in Table 3.3 is end-to-end and does not expose the same split.

| Data set | $m$ | Embedding | LibLinear | Pegasos | `snacks` |
|---|---|---|---|---|---|
| splice | 500 | 0.0 | 0.3 | 0.0 | 0.1 |
| mnist | 5 000 | 25.5 | 3.2 | 1.2 | 0.4 |
| w8a | 5 000 | 26.4 | 9.9 | 1.0 | 0.4 |
| ijcnn1 | 5 000 | 18.9 | 0.9 | 0.2 | 0.1 |
| epsilon | 2 000 | 62.0 | 25.3 | 14.7 | 1.8 |
| covtype | 5 000 | 53.3 | 3.3 | 1.5 | 0.1 |
| MSD | 5 000 | 60.2 | 14.2 | 9.5 | 0.8 |
| SUSY | 500 | 31.1 | 56.1 | 35.2 | 2.3 |
| HIGGS | 200 | 28.7 | 121.2 | 141.1 | 3.9 |

Our main findings are summarized below.

- **Accuracy is essentially unchanged.** The four methods remain within one percentage point of each other on every data set; `snacks` is best or tied-best on `w8a`, `ijcnn1`, `covtype`, MSD, and SUSY, and stays within 0.9 points of the best method on the remaining rows.
- **Runtime separates the methods.** At the same Nyström rank, `snacks` is the fastest or tied method on every data set.
- **The solver is no longer the bottleneck.** In Table 3.4, the `snacks` residual after embedding is at most 4 seconds; most of its reported training time is the shared Nyström embedding, for example 60.2 of 60.9 seconds on MSD and 28.7 of 32.6 seconds on HIGGS.
- **The speedup grows with scale.** On `covtype` and MSD, `snacks` trains in slightly less time than the next-fastest method (53.4 versus 54.8 seconds, and

60.9 versus 69.6 seconds); on SUSY and HIGGS it stays under 35 seconds versus 66.3 and 144.8 seconds for the next-fastest baseline.

### 3.5.3 Scaling with the number of samples

We compare how the four training methods scale as the embedded problem grows. The question is whether one method scales better or worse than the others with the training-problem dimension $n \times m$.

We use synthetic data to vary $n$ and $m$ directly while keeping the data-generating mechanism fixed. For each grid point $(n, m)$, we sample $n$ training points, select $m$ Nyström columns, form the embedded matrix $Z \in \mathbb{R}^{n \times r}$, and train the methods with the same protocol as above. We record wall-clock training time and plot it against $n \times m$ on logarithmic axes. Each point is the median over five runs.

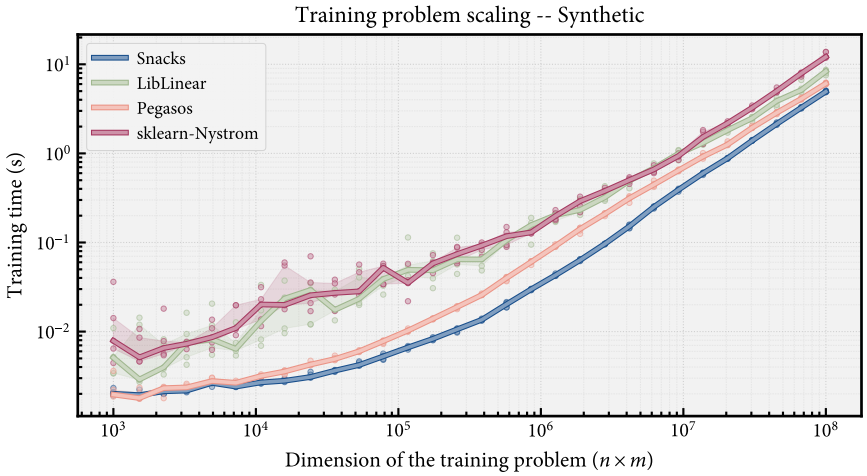



Figure 3.1: Training time versus the dimension of the training problem $n \times m$ (synthetic data).

Figure 3.1 shows the expected behavior: all methods scale approximately linearly with $n \times m$. The curves have nearly the same slope; `snacks` ties the best times on the smallest problems and is consistently faster on the larger ones.

### 3.5.4 Effect of the Nyström rank on cost and accuracy

We study how the Nyström rank $m$ affects training time and test error, comparing its influence on the latter with that of the regularization parameter $\lambda$. The goal is the smallest $m$ at which the test error has plateaued. For cost, we fix $10^6$ synthetic examples and vary $m$ from 10 to 1000 for `snacks` and Pegasos (Figure 3.2). Each

iteration acts on an $m$-dimensional vector, so at a fixed iteration count the training time should be affine in $m$: a term proportional to $m$ plus a rank-independent overhead.

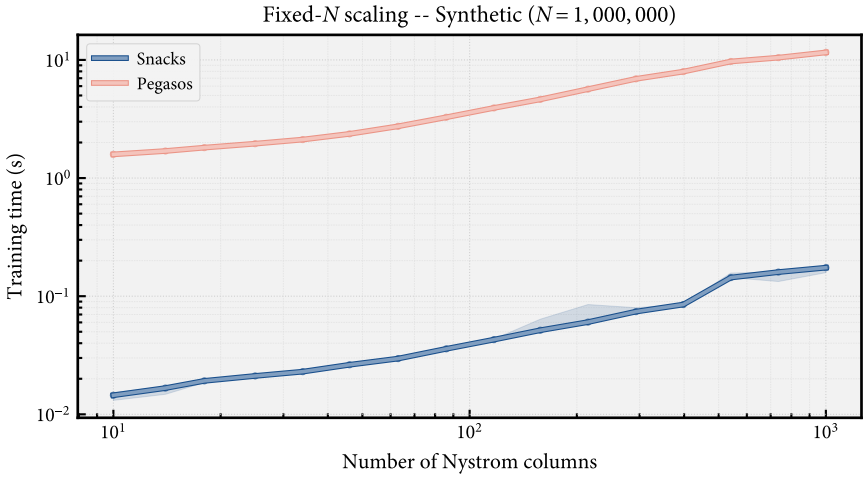



Figure 3.2: Training time of `snacks` and Pegasos against the Nyström rank $m$, at fixed $N = 10^6$. Both axes are logarithmic. Curves show the median over five runs and shaded bands the interquartile range.

Across the range, `snacks` is 65 to 110 times faster than Pegasos. At $m = 10$ the times are 0.015 s and 1.6 s; at $m = 117$, 0.043 s and 3.9 s; at $m = 1000$, 0.17 s and 11.6 s. Both curves grow slowly up to about a hundred columns and faster beyond, and both are consistent with an affine cost in $m$. The figure shows that the advantage of `snacks` stays roughly constant with $m$.

For accuracy, we use the `covtype` dataset and record the test error over $m \in [50, 1500]$ and $\lambda \in \left[10^{-4}, 3 \times 10^{-3}\right]$.

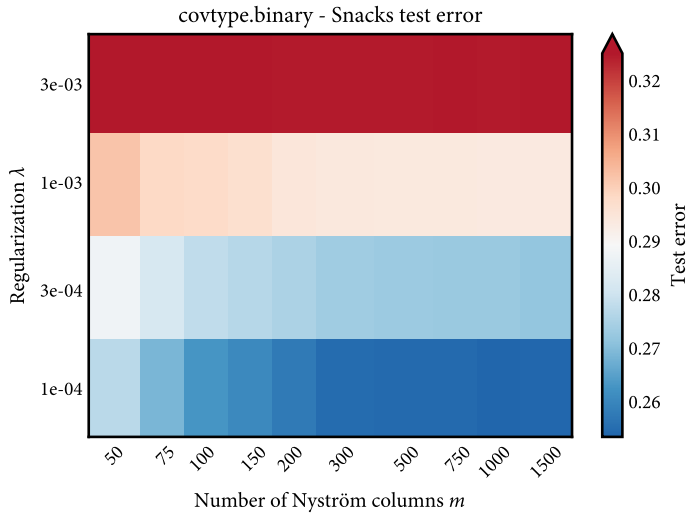



Figure 3.3: Test error of `snacks` on `covtype` over the Nyström rank $m$ and regularization parameter $\lambda$.

On this grid $\lambda$ dominates the test error, while $m$ matters only below a few hundred columns. At $m = 1500$, lowering $\lambda$ from $3 \times 10^{-3}$ to $10^{-4}$ cuts the error from 0.325 to 0.254. At $\lambda = 10^{-4}$, raising $m$ from 50 to 100 lowers it from 0.277 to 0.263, and to 0.255 at $m = 300$; beyond $m = 300$ it changes by less than 0.2 percentage point. The error plateaus in the same way at $\lambda = 3 \times 10^{-4}$ and $10^{-3}$, and at $\lambda = 3 \times 10^{-3}$ it is essentially independent of $m$. The smallest error on the grid occurs at $\lambda = 10^{-4}$ and $m = 1000$, but $m = 300$ already reaches nearly the same value. A rank of a few hundred therefore gives nearly the smallest test error, and larger ranks increase the training time with no measurable gain.

### 3.5.5 Effect of the Nyström stabilizers

We measure the effect of the truncation and of the ridge shift of the Nyström approximation separately. Each is compared with an unstabilized base that keeps every positive eigenvalue of $K_{J,J}$ and applies no shift. Truncation discards the eigenvalues below $10^{-4}\sigma_{max}$, and the shift eigendecomposes $K_{J,J} + 10^{-6}I_m$. For all variants, we fix the data split, the sampled columns, the kernel bandwidth, and the solver seed. The reported training time covers three steps: computing the Nyström approximation from the sampled columns, that is, the eigendecomposition of $K_{J,J}$ followed by the truncation or the shift; embedding the training and validation sets; and solving the resulting linear SVM problem (3.38) with RASSG-r.

Table 3.5: Effect of truncation and of the ridge shift relative to the unstabilized base, over five seeds. Rank is the embedding dimension of the base and of the truncated variant. Speedup is the ratio of median training times (base over variant), and $\Delta$ acc. is the change in test accuracy in percentage points (mean $\pm$ standard deviation).

| | | | Truncation | | Shift | |
|---|---|---|---|---|---|---|
| Dataset | $m$ | Rank | Speedup | $\Delta$ acc. (pts) | Speedup | $\Delta$ acc. (pts) |
| splice | 500 | $498 \to 493$ | $\times 0.5$ | $+1.09 \pm 0.54$ | $\times 1.1$ | $+0.73 \pm 1.04$ |
| mnist | 5000 | $5000 \to 1488$ | $\times 1.5$ | $-0.15 \pm 0.06$ | $\times 1.0$ | $0.00 \pm 0.00$ |
| w8a | 5000 | $4559 \to 1931$ | $\times 1.8$ | $+1.06 \pm 0.10$ | $\times 1.3$ | $+1.14 \pm 0.09$ |
| ijcnn1 | 5000 | $4297 \to 227$ | $\times 1.5$ | $-0.06 \pm 0.21$ | $\times 0.9$ | $+0.05 \pm 0.15$ |
| epsilon | 2000 | $2000 \to 1775$ | $\times 1.1$ | $-0.29 \pm 0.04$ | $\times 1.1$ | $+0.01 \pm 0.02$ |
| covtype | 5000 | $2998 \to 80$ | $\times 2.0$ | $-0.18 \pm 0.06$ | $\times 0.9$ | $-0.01 \pm 0.08$ |
| MSD | 5000 | $5000 \to 1014$ | $\times 1.9$ | $-0.29 \pm 0.12$ | $\times 1.0$ | $+0.07 \pm 0.08$ |
| SUSY | 500 | $500 \to 286$ | $\times 1.2$ | $-0.04 \pm 0.01$ | $\times 1.0$ | $0.00 \pm 0.01$ |
| HIGGS | 200 | $200 \to 200$ | $\times 1.0$ | $0.00 \pm 0.00$ | $\times 1.0$ | $+0.01 \pm 0.01$ |

Table 3.5 shows that truncation reduces the embedding dimension on every dataset except HIGGS, by up to a factor of 37 on covtype. It shortens training by a factor of 1.1 to 2.0 on the seven other datasets, while changing test accuracy by at most 0.3 percentage points, except on splice and w8a, where accuracy increases by about 1 point. On splice, whose training takes less than a second, truncation is slower by a factor of two. The shift leaves the embedding dimension unchanged and has a smaller effect: it changes training time by at most a factor of 1.3 and test accuracy by at most 0.1 percentage points, except on splice and w8a, where accuracy increases by 0.7 and 1.1 points.

### 3.5.6 Optimizer ablation

We next fix the Nyström representation and compare the two primal solvers on the same objective. This shows the effect of the more sophisticated RASSG-r updates against Pegasos (Shalev-Shwartz et al., 2011), which uses stochastic subgradient steps without restarts or proximal centering.

We train both solvers on six datasets and record test error as a function of elapsed solver time.

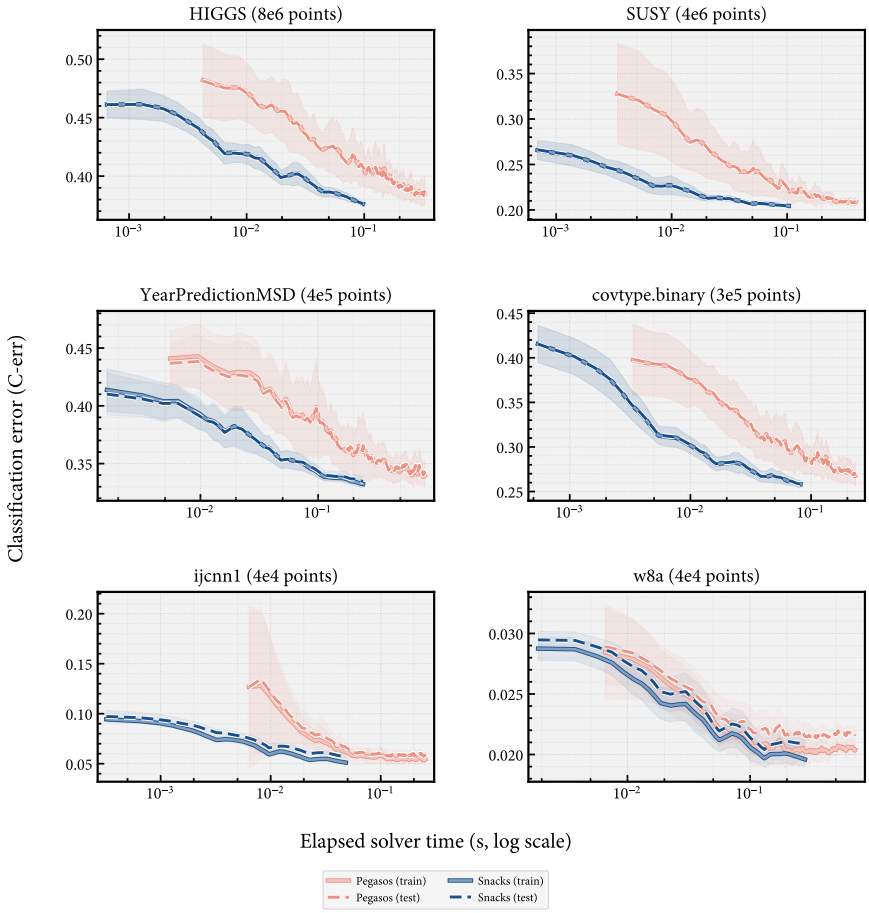



Figure 3.4: Test error as a function of solver time. The time axis is logarithmic.

In Figure 3.4, `snacks` reaches a given error level in less time than Pegasos, and both converge to a similar final error. The gap is widest on the largest datasets, HIGGS

and SUSY, and nearly closes on the smallest, ijcnn1 and w8a, highlighting the advantage of `snacks` on large problems.

RASSG-r has six hyperparameters — the schedule parameters S, J, $T_0$, $\gamma$, $c_\beta$, and $\rho_\beta$ — all fixed to the values in Table 3.1. To test sensitivity to these choices, we grid search them on covtype, ijcnn1, and mushrooms across a range of Nyström rank $m$ and regularization parameter $\lambda$, and record the gain in test accuracy over the default setting.

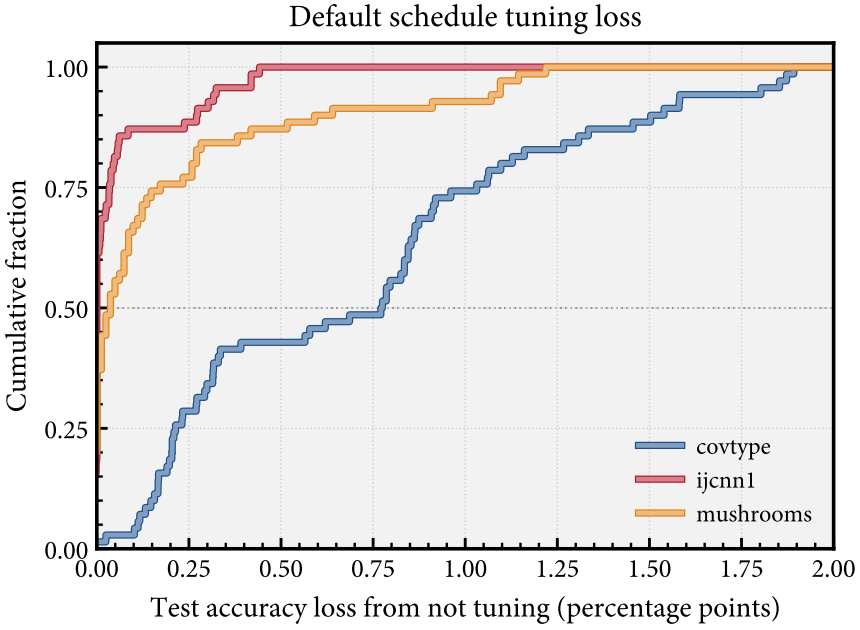



Figure 3.5: Empirical cumulative distribution function of the loss in test accuracy from using the default optimizer setting instead of the best tested setting for each pair $(m, \lambda)$.

Figure 3.5 shows that this gain concentrates near zero across the tested grid, so the default optimizer setting costs little accuracy over the tested grid.

### 3.5.7 Summary of empirical findings

The experiments support three conclusions. First, RASSG-r is the fastest or tied solver on every tested data set while remaining within 0.9 percentage points of the best test error. Second, the speed advantage grows with problem size and is consistent with the $O(nm)$ cost bound in Theorem 3.3. Third, the six schedule

parameters are insensitive to per-problem tuning: fixing them to the defaults costs negligible accuracy over the tested grid.

## 3.6 EXTENSIONS

Our solver `snacks` decouples the kernel approximation, the optimization method and the stochastic oracle used by the optimization method. This makes changing the loss or the regularizer easy as it suffices to change the oracle.

We can write the embedded objective in the general form:

$$F(u) = \frac{1}{n}\sum_{i=1}^{n} \ell\Big(y_i, z_i^\top u\Big) + \lambda\Omega(u), \quad u \in \mathbb{R}^r, \tag{3.55}$$

with loss $\ell$ and regularizer $\Omega$. Sampling an index $i \sim \mathrm{Unif}\{1, \dots, n\}$ gives the stochastic subgradient

$$g(u; i) = \lambda\, s_\Omega(u) + \partial_t \ell\Big(y_i, z_i^\top u\Big) z_i, \quad s_\Omega(u) \in \partial\Omega(u), \tag{3.56}$$

where $\partial_t \ell$ is the derivative of $\ell$ in its score argument. We now describe how to compute $g(u; i)$ for a set of large-margin losses and standard regularizers.

**Losses.** Apart from the hinge loss, which is the default in `snacks`, any loss that is Lipschitz in its score argument can be directly plugged. In particular, two common alternatives to the hinge loss are the logistic loss and the modified Huber loss. The logistic loss $\ell_{\log}(y, t) = \log\big(1 + e^{-yt}\big)$ is smooth and 1-Lipschitz, with $\partial_t \ell_{\log} = -y/\big(1 + e^{yt}\big)$. The modified Huber loss is a 4-Lipschitz, smoothed version of the hinge; it is defined as $\ell_{\mathrm{mh}}(y, t) = \begin{cases} \max(0, 1-yt)^2 & \text{if } yt \geqslant -1 \\ -4yt & \text{if } yt < -1 \end{cases}$, and has derivative $\partial_t \ell_{\mathrm{mh}}(y, t) = \begin{cases} 0 & \text{if } yt \geqslant 1 \\ -2y(1-yt) & \text{if } -1 \leqslant yt < 1 \\ -4y & \text{if } yt < -1 \end{cases}$. Figure 3.6 visualizes these three losses.

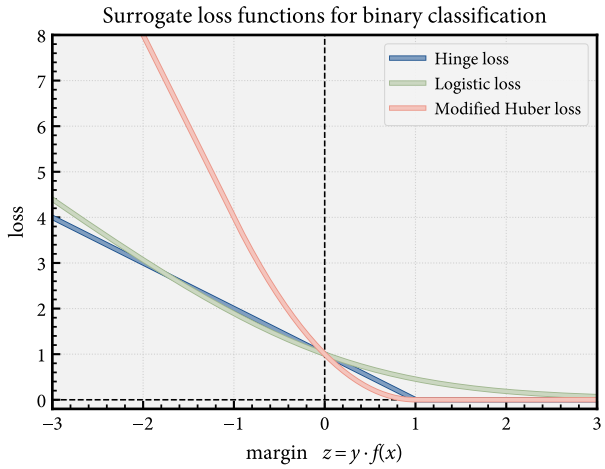



Figure 3.6: The hinge, logistic, and modified Huber losses as functions of the margin $yt$. All three are large-margin and Lipschitz in the score $t$, so each can replace the hinge loss in `snacks` by changing only the stochastic oracle (3.56).

**Regularizers.** Similarly, any convex regularizer can be plugged in `snacks` as long as we can compute a subgradient $s_\Omega(u)$. We describe the subgradient for the $\ell_1$ norm and the elastic net, which are common alternatives to the squared norm. The $\ell_1$ norm $\Omega(u) = \|u\|_1$ is nonsmooth and separable, with subdifferential given componentwise by

$$\partial|u_j| = \begin{cases} \{\operatorname{sign}(u_j)\} & \text{if } u_j \neq 0 \\ [-1, 1] & \text{if } u_j = 0 \end{cases}. \tag{3.57}$$

Using the convention $\operatorname{sign}(0) = 0$, a valid subgradient is therefore $s_\Omega(u) = \operatorname{sign}(u)$, taken componentwise, which favors sparse weights $u$. The elastic net $\Omega(u) = (1-\rho)\frac{1}{2}\|u\|_2^2 + \rho\|u\|_1$ combines the two. Its subdifferential is $\partial\Omega(u) = (1-\rho)u + \rho\,\partial\|u\|_1$, and we similarly use $s_\Omega(u) = (1-\rho)u + \rho\operatorname{sign}(u)$ for $\rho \in (0, 1)$.

**Convergence rates.** The analysis of the ASSG and RASSG methods in Xu et al. (2019) applies under a local growth condition on $F$ around its minimizers $F(u) - F^* \geqslant c\,\operatorname{dist}(u, \mathcal{U}^*)^\theta$ for some $c > 0$, where $\theta \geqslant 1$ is the growth exponent and $\mathcal{U}^*$ is the set of minimizers. The convergence is given in high probability by $\tilde{\mathcal{O}}\big(\varepsilon^{-2(1-1/\theta)}\big)$ for ASSG, so a smaller $\theta$, which depends on both the loss and the regularizer, gives a faster rate. As soon as the objective is strongly convex (whenever the regularizer is

strongly convex), we have $\theta = 2$ and the rate is $\tilde{\mathcal{O}}(1/\varepsilon)$ as in Theorem 3.3. However, for the hinge loss with the $\ell_1$ norm, the objective is polyhedral and we have $\theta = 1$, which gives a much faster rate, $\tilde{\mathcal{O}}(\log(1/\varepsilon))$.

**Remark.** The $\theta = 1$ rate above holds because polyhedral convex functions have weak sharp minima, i.e. they satisfy the growth condition with $\theta = 1$. In this case the minimizer is not unique and the constant $c$ depends on the data, so we get a better asymptotic rate but not necessarily better constants.

## 3.7 Conclusion

`snacks` solves kernel SVMs on large datasets by combining a Nyström embedding with RASSG-r, a restarted accelerated stochastic subgradient method. The experiments show it to be the fastest or tied solver on every dataset tested, by a margin that widens as the number of samples $n$ grows. Its test error stays within a fraction of a percentage point of the best baseline. Moreover, the six hyperparameters of the optimizer need no per-problem tuning. At large $n$ the dominant cost is forming and storing the feature matrix, whose number of columns is the Nyström rank $m$. Forming it block by block, or storing it in lower precision, would reduce this cost and is an interesting research direction; related memory gains have been demonstrated for binarized kernel sketches (Delogne & Jacques, 2025). Either route would extend the method to even larger datasets without changing the optimizer.

# 4 | DUAL METHODS FOR CONVEX HULL PRICING


## ABSTRACT

We study first-order methods for computing convex hull prices in electricity markets. Convex hull pricing is a pricing rule designed to handle markets with non-convexities such as startup costs and minimum up and down times, and it aims at minimizing total lost opportunity costs, which include both actual losses and missed profit opportunities. Computing these prices reduces to maximizing a partial Lagrangian dual of the original unit-commitment mixed-integer program, where power balance constraints are dualized. The dual objective to minimize is a sum of non-smooth convex functions, one per generator, whose subgradient is obtained by solving one mixed-integer program per generator. The cost of these per-generator programs, together with the slow convergence inherent to non-smooth optimization, makes the choice of method decisive at the scale of real markets.

We benchmark a large set of dual first-order methods for this problem, most of which had not previously been applied to convex hull pricing. Among them, we introduce variable-metric multi-cut bundle methods, which exploit the separability of the dual by maintaining a separate polyhedral model for each group of generators and restricting each step through an adaptive diagonal metric. On two real-world instance sets, the retained variable-metric bundle variants are the strongest methods under the 15-minute time limit, while on the Californian instances subgradient and smoothing-based methods reach the target accuracy about as fast.


## BASED ON

*This chapter is based on Tanji, S., Kamri, Y., Glineur, F., & Madani, M. (2025). Dual first-order methods for efficient computation of convex hull prices.* Arxiv Preprint Arxiv:2504.01474.*, which is currently undergoing a second round of review, with minor revisions requested, at the European Journal of Computational Optimization (EJCO).*

## IN THIS CHAPTER

The first section of this chapter (Section 4.1) reviews convex hull pricing in electricity markets and the existing primal and dual approaches to computing it. In Section 4.2, we review the derivation of primal and dual formulations for computing CH prices. In Section 4.3, we introduce and describe a set of eight first-order methods to solve the dual formulation. In Section 4.5, we conduct extensive numerical experiments to compare all presented methods on real-world instances. In Section 4.6, we conclude with our recommendations among all presented dual methods.

## 4.1 BACKGROUND

### 4.1.1 Pricing in electricity markets

**Electricity markets.** Over the past decades, the global electricity system has shifted from centrally planned, bundled operations to a regulated market-based structure. In such markets, a system operator is in charge of balancing generation and demand while meeting the technical constraints of producers and maximizing social welfare. This scheduling problem is known as *Unit Commitment* (UC) in the power systems literature. However, market equilibrium prices exist if and only if there is no duality gap between the primal UC problem and its Lagrangian dual in which balance conditions have been dualized, which is in general not the case in power markets. Indeed, producers face technical constraints such as startup costs or minimum up and down times that can only be modeled with binary variables, rendering the feasible region of the optimization problem non-convex. This is a well-known fact, and basic examples illustrating typical market equilibrium impossibilities were given, for example, in Scarf (1994), using an electricity generation context. While other interesting approaches relying on marginal pricing combined with minimum-income conditions for generators (or their analogue on the demand side) have been considered in the literature (see for instance Fernández-Blanco et al., 2015; Madani & Van Vyve, 2017, and references therein), convex hull pricing also has the advantage of decoupling the computation of the allocation from the computation of market prices.

**Convex hull pricing.** Due to the general impossibility of having uniform prices supporting a competitive equilibrium in power systems, pricing schemes in electricity markets often combine a uniform electricity price and side payments (or uplifts) to producers. These side payments are paid when no market equilibrium exists, i.e. when producers face missed profits and lack incentive to stay in the market. The convex hull (CH) pricing approach, first introduced in Ring (1995) and further explored in Gribik et al. (2007), computes uniform prices that minimize the producers' lost opportunity costs[5]. Because these side payments measure the deviation from a market equilibrium, CH prices are the uniform prices with the smallest total uplift caused by nonconvexities. Recent work compares the short- and long-term economic properties of CH pricing with those of existing pricing rules

[5] By lost opportunity costs, we mean the difference between the profit a producer could earn by self-scheduling at those prices and the profit assigned by the market schedule, truncated at zero. In this chapter and for ease of exposition, we will use interchangeably the terms total side payment, total uplift and total lost opportunity cost, although nuances exist.

and shows several advantages of CH pricing; see Byers & Hug (2023) and Stevens et al. (2024).

It was first observed in Gribik et al. (2007) that computing CH prices is equivalent to solving the Lagrangian dual associated with the main UC problem with dualized balance constraints, or more generally the system-wide constraints.

**Computational hardness of computing CH prices.** As stated above, computing CH prices exactly amounts to solving the partial Lagrangian relaxation of the Unit Commitment problem. This is in general computationally expensive as it requires maximizing a nonsmooth concave function of the dual prices for which a supergradient can be computed by solving a Mixed Integer Linear Program (MILP) for each generator; see Section 4.2.2 for details on this approach. Approximate convex hull prices are often calculated using the continuous relaxation of the original UC problem. This is done, for instance, by the US ISO PJM, a regional system operator (PJM Interconnection, 2017). The continuous relaxation yields exact convex hull prices when the convex hull of each generator's feasible set is given by its continuous relaxation. For example, the Lagrangian multipliers of the balance condition in the continuous relaxation (of the binary variables) of the UC problem are exactly the CH prices when the technical constraints are limited to minimum up/down times and constant start-up (or shut-down) costs, see for example Hua & Baldick (2016) and references therein. In short, in those setups, a linear program of the size of the original UC problem is sufficient to compute the CH prices. When only ramp conditions are considered, it can be shown that a similar approach can be followed, see Madani et al. (2018). In the more realistic setting considered here, the model includes the standard technical constraints used in most unit-bidding markets, in particular minimum up and down times and ramp constraints. Then the continuous relaxation of the generators' feasible sets is not tight, and the continuous relaxation of the primal problem does not yield exact CH prices. Computing CH prices in this setting is the subject of this chapter.

### 4.1.2 Computational approaches for CH pricing

**Primal approaches.** First consider primal approaches where one calculates CH prices by solving a convexified primal UC problem in which each producer's feasible set is replaced by its convex hull. Extended formulations have been leveraged in such approaches, where the convex hulls are described as the projection of higher-dimensional polytopes, see Knueven et al. (2022) and Yu et al. (2020). In Knueven et al. (2022), the large-scale extended formulation is tackled via Benders decomposition, while Yu et al. (2020) dynamically add extended formulations only for the generators for which they are needed. Paquet (2021) independently derives

a refined version of the approach in Yu et al. (2020). Instead of including the full extended formulation, it dynamically adds, for each generator, only the parts corresponding to the intervals during which that generator is on. This refinement emerges naturally as an application of a general column generation procedure to the extended formulations of Sadykov & Vanderbeck (2013). Finally, Andrianesis et al. (2021) use a standard Dantzig-Wolfe decomposition method, which can also be viewed as leveraging extended formulations, where the underlying extended formulations consist in describing the convex hulls using their vertex representation. Stabilized decomposition methods have also received growing attention on the primal side. Zhang et al. (2025) and Pecci & Jenkins (2025) develop regularized and stabilized Benders decompositions for capacity expansion and energy system planning, addressing oscillation issues closely related to those motivating bundle level methods. In a related direction, Zhang et al. (2024) propose a dynamic oracle adaptation strategy that adjusts stabilization parameters adaptively, a concept analogous to the dynamic level parameter we study in Section 4.3.

**Dual approaches.** It is well known that Dantzig-Wolfe decompositions applied to the primal formulation correspond to applying Kelley's cutting plane algorithm to the Lagrangian dual directly in the dual space of market prices (see Briant et al., 2008 for the link between both methods). However, Kelley's method (and therefore its primal counterparts, the Dantzig-Wolfe decompositions) suffer from instability in the sense that near-optimal iterates can lead to the following iterates being much further away from the optimum, even on simple problems (see e.g. (Nesterov, 2004, Example 3.3.1)). This motivated bundle methods which are cutting-plane algorithms with stabilization techniques, see Frangioni (2020) and Lemaréchal et al. (1995) for an extensive discussion and review of the literature on such methods. In Stevens & Papavasiliou (2022), the authors solve the CH pricing problem via a Bundle Level Method and report numerically that this approach is competitive on real-world market instances. To the best of our knowledge, the existing literature does not report on other dual first-order methods for CH pricing, beyond the classic subgradient and level methods.

## 4.2 Unit commitment and convex hull pricing formulations

In electricity markets, the market operator ensures that total generation meets the demand under a market clearing constraint over a given time horizon, while each participant aims to maximize their profit. This decision-making problem, the unit commitment problem, is described in Section 4.2.1. In Section 4.2.2, we present the

CH pricing problem as a Lagrangian dual problem of the UC problem where the power balance constraints have been dualized.

### 4.2.1 The UC problem

For simplicity, we write a general UC problem; see Knueven et al. (2020) for more information on formulations of unit commitment problems.

Let $T$ denote the number of time periods, and let $\mathcal{T} = \{1, ..., T\}$ denote the set of periods. Let $\mathcal{G}$ denote the set of generators, $L_t$ the inelastic demand at time $t$, and $l_t$ the served demand at time $t$, with $0 \leqslant l_t \leqslant L_t$. As demand is treated as inelastic, we should also consider how non-served energy penalizes total welfare: let $C_{VOLL}$ be the unit cost of non-served energy, also called the value of lost load.

For each generator $g$ and period $t$, let $p_t^g$ be power output, $u_t^g$ the binary commitment decision, $v_t^g$ the binary startup decision, and $w_t^g$ the binary shutdown decision. We write $p^g = \left(p_t^g\right)_{t\in\mathcal{T}}$, $u^g = \left(u_t^g\right)_{t\in\mathcal{T}}$, $v^g = \left(v_t^g\right)_{t\in\mathcal{T}}$, and $w^g = \left(w_t^g\right)_{t\in\mathcal{T}}$ for generator schedules. We call $p^g$ the *production part* of a schedule $(p^g, u^g, v^g, w^g)$. The stacked vectors over all generators are denoted by $\mathbf{p}, \mathbf{u}, \boldsymbol{v}$, and $\boldsymbol{w}$, and the served-load vector is $\mathbf{l} = (l_t)_{t\in\mathcal{T}}$.

For generator $g$, the period cost is

$$C^g\left(p_t^g, u_t^g, v_t^g\right) = C_M^g p_t^g + C_{NL}^g u_t^g + C_F^g v_t^g. \tag{4.1}$$

Here $C_M^g$ is the marginal cost, $C_F^g$ is the startup cost, and $C_{NL}^g$ is the no-load cost. The objective is independent of the shutdown schedule $w^g$ except through feasibility.

To define feasibility, let $\underline{P}^g$ and $\overline{P}^g$ be the minimum and maximum running capacities, $UT^g$ and $DT^g$ the minimum up and down times, $SU^g$ and $SD^g$ the startup and shutdown levels, and $RU^g$ and $RD^g$ the ramp-up and ramp-down limits. The ramp limits bound the maximum allowable increase or decrease in power output between two consecutive time periods. The feasible schedule set $\mathcal{P}^g$ is defined by

$$\begin{cases} u_t^g - u_{t-1}^g = v_t^g - w_t^g & \forall t \in \{2, ..., T\}, \\ \sum_{i=t-UT^g+1}^{t} v_i^g \leqslant u_t^g & \forall t \in [UT^g, T], \\ \sum_{i=t-DT^g+1}^{t} w_i^g \leqslant 1 - u_t^g & \forall t \in [DT^g, T], \\ \underline{P}^g u_t^g \leqslant p_t^g \leqslant \overline{P}^g u_t^g & \forall t \in \mathcal{T}, \\ p_t^g - p_{t-1}^g \leqslant RU^g u_{t-1}^g + SU^g v_t^g & \forall t \in \{2, ..., T\}, \\ p_{t-1}^g - p_t^g \leqslant RD^g u_t^g + SD^g w_t^g & \forall t \in \{2, ..., T\}, \\ \left(u_t^g, v_t^g, w_t^g\right) \in \{0, 1\}^3 & \forall t \in \mathcal{T} \end{cases} \tag{4.2}$$

The UC decision-making problem is then

**Problem 4.1.** (Unit Commitment)

$$\min_{(p,u,v,w,l)} F(p,u,v,w,l) := \sum_{\substack{t\in\mathcal{T}\\ g\in\mathcal{G}}} C^g\big(p_t^g, u_t^g, v_t^g\big) + \sum_{t\in\mathcal{T}} C_{\text{VOLL}}(L_t - l_t) \tag{4.3}$$

subject to:

$$\sum_{g\in\mathcal{G}} p_t^g = l_t \quad \forall t \in \mathcal{T}, \tag{4.4}$$

$$(p^g, u^g, v^g, w^g) \in \mathcal{P}^g \quad \forall g \in \mathcal{G}, \tag{4.5}$$

$$0 \leqslant l_t \leqslant L_t \quad \forall t \in \mathcal{T}. \tag{4.6}$$

### 4.2.2 Formulations for CH pricing

Convex hull prices are usually defined as the prices minimizing uplifts (or side payments, see the paragraph on convex hull pricing above). In this work, we use instead the following two alternate definitions, proposed and shown to be equivalent in Geoffrion (1974).

**Proposition 4.1.** *The following characterizations are equivalent:*

1. *Convex hull prices are the optimal variables of the Lagrangian dual associated with the UC Problem in which one dualizes the power-balance constraint (4.4).*
2. *Convex hull prices are the dual variables $\pi^{CH}$ associated with the power-balance constraint (4.4) of Problem 4.1 where sets $\mathcal{P}^g$ have been replaced by their convex hulls $\text{conv}(\mathcal{P}^g)$.*

This equivalence is a well-established result in integer programming duality. In the context of convex hull pricing, Hua & Baldick (2016) generalized this relationship to accommodate specific nonlinear objective functions. We use the first characterization above. We only relax the power-balance constraint (4.4) because it is the only constraint coupling the generators. This yields the following Lagrangian function of Lagrange multipliers $\pi_t$, corresponding to the electricity price at time period $t$:

$$\begin{aligned} \mathcal{L}(\pi) = \min_{p,u,v,w,l} & F(p,u,v,w,l) - \sum_{t\in\mathcal{T}} \pi_t \left( \sum_{g\in\mathcal{G}} p_t^g - l_t \right) \\ \text{subject to: } & (p^g, u^g, v^g, w^g) \in \mathcal{P}^g \quad \forall g \in \mathcal{G}, \\ & 0 \leqslant l_t \leqslant L_t \quad \forall t \in \mathcal{T}. \end{aligned} \tag{4.7}$$

CH prices $\pi^{CH}$ are the maximizers of $\mathcal{L}(\pi)$. In the computations below, we impose a bounded admissible price box Q, corresponding to market price range restrictions, and therefore compute CH prices under these imposed bounds. After rearranging terms in (4.7), we obtain the following Lagrangian relaxation problem, (partial) dual of UC:

$$\pi^* \in \arg\max_{\pi \in Q} \mathcal{L}(\pi), \quad \mathcal{L}(\pi) = \mathcal{L}_0(\pi) + \sum_{g \in \mathcal{G}} \mathcal{L}_g(\pi). \tag{4.8}$$

The demand component is

$$\begin{aligned} \mathcal{L}_0(\pi) = \min_{l} \sum_{t \in \mathcal{T}} [C_{VOLL}(L_t - l_t) + \pi_t l_t] \\ \text{subject to: } 0 \leqslant l_t \leqslant L_t \quad \forall t \in \mathcal{T}, \end{aligned} \tag{4.9}$$

and, for each $g \in \mathcal{G}$, the generator component is

$$\begin{aligned} \mathcal{L}_g(\pi) = \min_{(p^g, u^g, v^g, w^g)} \sum_{t \in \mathcal{T}} \left[ C^g\left(p_t^g, u_t^g, v_t^g\right) - \pi_t p_t^g \right] \\ \text{subject to: } (p^g, u^g, v^g, w^g) \in \mathcal{P}^g. \end{aligned} \tag{4.10}$$

This formulation is separable, meaning we can consider the pricing problem one generator at a time, allowing considerable computational benefits.

For a price vector $\pi$, let $p_g^*(\pi)$ and $l^*(\pi)$ denote optimal production and demand solutions of (4.10) and (4.9). Because each component objective is affine in $\pi$ after fixing a primal solution, a classical result gives the following oracle; see (Conforti et al., 2014, Corollary 8.3).

**Lemma 4.2.** (First-order oracle) *The dual function $\mathcal{L}$ is nonsmooth, concave, and piecewise affine. For a given $\pi$, one supergradient is*

$$l^*(\pi) - \sum_{g \in \mathcal{G}} p_g^*(\pi) \in \hat{\partial}\mathcal{L}(\pi). \tag{4.11}$$

*Consequently, for the convex function $f = -\mathcal{L}$, one subgradient is*

$$\sum_{g \in \mathcal{G}} p_g^*(\pi) - l^*(\pi) \in \partial f(\pi). \tag{4.12}$$

The minimization problem in $\mathcal{L}_0(\pi)$ can be solved in closed form.

**Lemma 4.3.** *We have*

$$l_t^*(\pi) = L_t \times \mathbb{1}_{\{\pi_t < C_{VOLL}\}} \quad \forall t \in \mathcal{T} \tag{4.13}$$

$$\mathcal{L}_0(\pi) = \sum_{t \in \mathcal{T}} L_t[C_{\text{VOLL}} - \max\{0, C_{\text{VOLL}} - \pi_t\}] \tag{4.14}$$

*Proof.* Equation (4.13) has a simple interpretation: to solve (4.9), one consumes as much as possible if $C_{\text{VOLL}} > \pi_t$ and nothing if $C_{\text{VOLL}} < \pi_t$, while any value $l_t \in [0; L_t]$ is optimal when $C_{\text{VOLL}} = \pi_t$. □

The constant term $\sum_{t \in \mathcal{T}} L_t C_{\text{VOLL}}$ can be removed from (4.14), (4.3), and (4.9) as far as finding optimal solutions is concerned.

The dual CH pricing problem (4.8) is a concave maximization problem, but solving it is equivalent to minimizing the nonsmooth convex objective

$$f(\pi) = -\mathcal{L}(\pi). \tag{4.15}$$

We use the symbol $f$ throughout the methods and experiments. At a point $\pi$, the oracle returns the value $f(\pi)$ together with the subgradient $g(\pi) = \sum_{g \in \mathcal{G}} p_g^*(\pi) - l^*(\pi)$.

## 4.3 First-order methods

Many first-order methods solve nonsmooth convex optimization problems with convergence guarantees. Using such a method for CH pricing requires one first-order oracle call at each iteration; by Lemma 4.2 this amounts to solving one demand subproblem and $|\mathcal{G}|$ generator MILPs in (4.10), for example with a mixed-integer solver. We leave the discussion of stopping criteria to Section 4.5.4. We follow the common approach, used for example in Stevens & Papavasiliou (2022), of setting the admissible prices to the box $Q = [\pi_{\min}, \pi_{\max}]^T$. Most methods presented below are new in the context of computing CH prices, apart from the bundle level method (Algorithm 4.4) proposed in Stevens & Papavasiliou (2022) and the subgradient method with vanishing step lengths (Algorithm 4.1) proposed in Ito et al. (2013).

### 4.3.1 Subgradient-based methods

Projected subgradient methods (Boyd et al., 2003) are the simplest iterative algorithms for solving nonsmooth convex minimization problems.

**Algorithm 4.1 (Subgradient method (vanishing step lengths))**:

**Parameters:** initial step length $\eta$
**Inputs:** initial iterate $\pi^1 \in Q$
**repeat for** $k = 1, 2, \ldots$
Select $g^k \in \partial f(\pi^k)$.
Compute $\pi^{k+1} = P_Q\left(\pi^k - \frac{\eta}{k}\frac{g^k}{\|g^k\|}\right)$.
**until** a stopping criterion is satisfied.
Set $k_{best} \in \arg\min_{1 \leqslant i \leqslant k} f(\pi^i)$ **and** output $\pi^{k_{best}}$.

We use normalized subgradients and therefore call $\frac{\eta}{k}$ a step length. This presents the theoretical advantage of not requiring knowledge of an upper bound on the subgradient norm. One could also use a step length schedule decreasing as $\frac{\eta}{\sqrt{k}}$, with similar results. The projected subgradient method (Algorithm 4.1) is one of the earliest approaches to compute CH prices; see Ito et al. (2013). The method is, however, notoriously slow and hard to tune as its performance is significantly influenced by the choice of both the initial iterate $\pi^1$ and the stepsize sequence, i.e. initial step length $\eta$.

With regard to the initial iterate, we propose a warm-start strategy in Section 4.5.1.2. With regard to the stepsize sequence, B. Polyak proposed in Polyak (1963) the step size

$$h_k = \frac{f(\pi^k) - f(\pi^*)}{\|g^k\|^2}. \tag{4.16}$$

As the value $f(\pi^*)$ is not known in practice, an existing idea (see Boyd et al., 2003) is to approximate it using the lowest objective value so far $f^k_{best} = \min_{1 \leqslant i \leqslant k} f(\pi^i)$ and an estimate of the gap $\frac{\alpha}{k}$, which yields the following algorithm.

**Algorithm 4.2 (Subgradient method (estimated Polyak step sizes))**:

**Parameters:** initial suboptimality estimate $\alpha > 0$
**Inputs:** initial iterate $\pi^1 \in Q$
**repeat for** $k = 1, 2, ...$
Select $g^k \in \partial f(\pi^k)$.
Set $f^k_{\text{best}} = \min_{1 \leqslant i \leqslant k} f(\pi^i)$.
Compute $\pi^{k+1} = P_Q\left(\pi^k - \left(f(\pi^k) - \left(f^k_{\text{best}} - \frac{\alpha}{k}\right)\right)\frac{g^k}{\|g^k\|^2}\right)$.
**until** a stopping criterion is satisfied.
Set $k_{\text{best}} \in \arg\min_{1 \leqslant i \leqslant k} f(\pi^i)$ **and** output $\pi^{k_{\text{best}}}$.

We also investigate the optimal variant of the subgradient method recently proposed by Zamani & Glineur (2023), with a specific linearly decaying step length schedule which achieves the best possible last-iterate accuracy after a given fixed number of iterations.

**Algorithm 4.3 (Last-iterate optimal subgradient method)**:

**Parameters:** upper bound $R$ such that $\|\pi^1 - \pi^*\| \leqslant R$
**Inputs:** initial iterate $\pi^1 \in Q$, number of iterations $N$
**For** $k = 1, 2, ..., N$ perform the following steps:
Select $g^k \in \partial f(\pi^k)$.
Compute $\pi^{k+1} = P_Q\left(\pi^k - \frac{R(N+1-k)}{\sqrt{(N+1)^3}}\frac{g^k}{\|g^k\|}\right)$.
**Output:** last iterate $\pi^{N+1}$.

### 4.3.2 Parameter-free methods

Parameter-free methods are methods which do not require knowledge of the problem parameters to run, such as an upper bound on the subgradient norm.

Bundle level methods Bundle level methods (Lemaréchal et al., 1995) are arguably among the first *parameter-free* algorithms for nonsmooth convex optimization. The idea of bundle methods is to aggregate subgradient information in a bundle to maintain a polyhedral approximation of the objective function. In the CH pricing context, $f$ is piecewise affine, so each cut is exact on at least one face of the epigraph. This bundle approach differs considerably from previously considered algorithms where at each iteration, the current iterate is updated using only the

subgradient at the previous point. The term "level" refers to how new iterates are computed: the method projects onto a level set of the bundle model rather than taking a step along only the newest subgradient.

We present two variants: the Bundle Level Method (BLM), introduced in Lemaréchal et al. (1995) and first used for CH pricing in Stevens & Papavasiliou (2022), and the Bundle Proximal Level Method (BPLM), also introduced in Lemaréchal et al. (1995). Let $m_k(\pi)$ denote the current bundle model,

$$m_k(\pi) = \max_{1 \leqslant i \leqslant k} \left\{ f(\pi^i) + \langle g^i, \pi - \pi^i \rangle \right\}. \tag{4.17}$$

**Algorithm 4.4 (Bundle Level Method (BLM))**:

**Parameters:** level parameter $\alpha \in (0, 1)$.
**Inputs:** initial iterate $\pi^1 \in Q$
Initialize $UB_0 = +\infty$ **and** $LB_0 = -\infty$.
**repeat for** $k = 1, 2, \ldots$
Compute the oracle $\left(f(\pi^k), g^k \in \partial f(\pi^k)\right)$.
Set $UB_k = \min\left\{UB_{k-1}, f(\pi^k)\right\}$.
Set $\widehat{LB}_k = \min_{\pi \in Q} m_k(\pi)$ **and** $LB_k = \max\left\{LB_{k-1}, \widehat{LB}_k\right\}$.
Set the level value $\ell_k = LB_k + \alpha(UB_k - LB_k)$.
Set the level set $S_k = \{\pi \in Q : m_k(\pi) \leqslant \ell_k\}$.
Compute $\pi^{k+1} \in \arg\min_{\pi \in S_k} \frac{1}{2}\|\pi - \pi^k\|^2$.
**until** a stopping criterion is satisfied.
Set $k_{best} \in \arg\min_{1 \leqslant i \leqslant k} f(\pi^i)$ **and** output $\pi^{k_{best}}$.

**Algorithm 4.5 (Bundle Proximal Level Method (BPLM))**:

**Parameters:** level parameter $\alpha \in (0, 1)$.
**Inputs:** initial iterate $\pi^1 \in Q$
Initialize $UB_0 = +\infty$, $LB_0 = -\infty$, $\Delta_0 = +\infty$, **and** $\ell'_0 = +\infty$.
**repeat for** $k = 1, 2, \ldots$
Compute the oracle $\left(f(\pi^k), g^k \in \partial f(\pi^k)\right)$.
Set $UB_k = \min\left\{UB_{k-1}, f(\pi^k)\right\}$.
Set $\widehat{LB}_k = \min_{\pi \in Q} m_k(\pi)$ **and** $LB_k = \max\left\{LB_{k-1}, \widehat{LB}_k\right\}$.
Set $\ell_k = LB_k + \alpha(UB_k - LB_k)$.
Set $\ell'_k = \begin{cases} \min\{\ell_k, \ell'_{k-1}\} & \text{if } UB_k - LB_k \geqslant (1-\alpha)\Delta_{k-1}, \\ \ell_k & \text{otherwise.} \end{cases}$
Set $\Delta_k = \begin{cases} \Delta_{k-1} & \text{if } UB_k - LB_k \geqslant (1-\alpha)\Delta_{k-1}, \\ UB_k - LB_k & \text{otherwise.} \end{cases}$
Set $S'_k = \left\{\pi \in Q : m_k(\pi) \leqslant \ell'_k\right\}$.
Set $\hat{\pi}^k \in \arg\min_{1 \leqslant i \leqslant k} f(\pi^i)$.
Compute $\pi^{k+1} \in \arg\min_{\pi \in S'_k} \frac{1}{2}\|\pi - \hat{\pi}^k\|^2$.
**until** a stopping criterion is satisfied.
Set $k_{best} \in \arg\min_{1 \leqslant i \leqslant k} f(\pi^i)$ **and** output $\pi^{k_{best}}$.

Both bundle level algorithms make one oracle call, solve an LP to update the lower bound, and solve a quadratic program (QP) to project onto the level set. The stored lower bound $LB_k$ is nondecreasing by construction. The difference lies in the level set computation: in BPLM (Algorithm 4.5), the level set cannot increase until the gap $\Delta$ decreases substantially, whereas in BLM (Algorithm 4.4), it is updated at every iteration. This is illustrated in Figure 4.1. If cuts are deleted, the current model bound $\widehat{LB}_k$ may decrease, but the stored bound $LB_k = \max\left\{LB_{k-1}, \widehat{LB}_k\right\}$ remains nondecreasing. An interesting advantage of both BLM and BPLM is that the gap $\Delta_k = UB_k - LB_k$ provides an estimate of the current accuracy. This gap will be used in Section 4.5.1.1 to compute near-optimal convex hull prices.

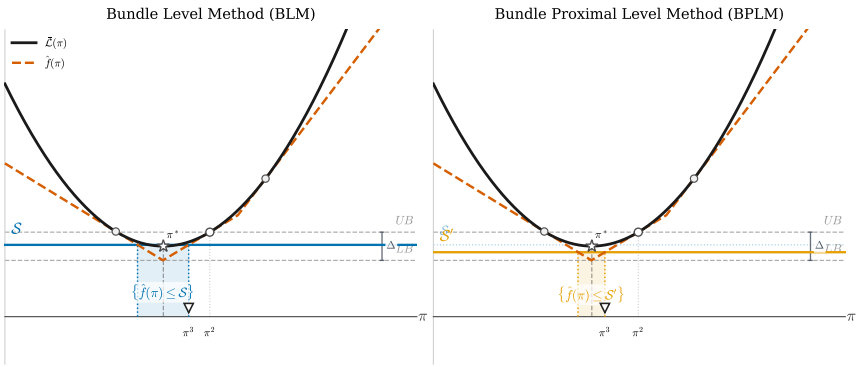



Figure 4.1: Comparison of BLM (left) and BPLM (right) on minimizing $f(\pi) = 0.3(\pi - 2.5)^2 + 1$, with $\pi^0 = 1.75$ and $\alpha = 0.55$. Both panels depict iteration $k = 2$, the first where the two updates differ. Because BPLM uses the tighter level set $\mathcal{S}'$, it produces an update closer to $\pi^*$: $\left|\pi^3_{\text{BPLM}} - \pi^*\right| = 0.207$ versus $0.400$ for BLM.

**Multicut bundle methods.** The bundle methods considered above are single-cut methods: at each oracle call, the $|\mathcal{G}|$ subproblems in (4.10) are aggregated into a single cut by summing subgradients of the individual functions $f_g$, thereby obtaining one subgradient of $f$. Because $f = f_0 + \sum_{g\in\mathcal{G}} f_g$ is separable, an alternative is to maintain a distinct polyhedral model for each component $f_g$. Each oracle call then adds $|\mathcal{G}|$ cuts, one per generator, resulting in a tighter lower approximation for the same oracle information. At an evaluated price $\pi^i$, let $l^i$ be a demand minimizer and $p^i_g$ a production minimizer for generator $g$. A subgradient of $f_0$ is $-l^i$, and a subgradient of $f_g$ is $p^i_g$. The generator-wise multicut master is therefore

$$\min_{\pi\in Q,\theta_0,(\theta_g)_{g\in\mathcal{G}}} \quad \theta_0 + \sum_{g\in\mathcal{G}} \theta_g \tag{4.18}$$

$$\text{subject to} \quad f_0(\pi^i) + \langle -l^i, \pi - \pi^i\rangle \leqslant \theta_0 \quad \forall i, \tag{4.19}$$

$$f_g(\pi^i) + \langle p^i_g, \pi - \pi^i\rangle \leqslant \theta_g \quad \forall g \in \mathcal{G}, \forall i. \tag{4.20}$$

When the upper price bound is strictly below $C_{\text{VOLL}}$, the demand component is affine up to the discarded constant and can be included exactly instead of through stored cuts. We denote the generator-wise variants by MC-BLM and MC-BPLM. MC-BLM is mentioned in Stevens & Papavasiliou (2022), where it is reported to improve performance.

In practice, this approach hardly scales to large instances because the number of cuts grows linearly with the number of generators and the number of iterations. We address this issue in 4.5.2.

**Variable-metric bundle methods.** In BLM and BPLM, the projection step is computed over the whole box Q. We also consider a diagonal variable metric in this step, with one radius $\delta_t^k$ per time period $t \in \mathcal{T}$. Let $|\pi|^k$ be the projected point, that is, $\pi^k$ in BLM and $\hat{\pi}^k$ in BPLM. The metric replaces the box Q, in the projection step and in the lower bound, by the trust region

$$X_k = \left\{\pi \in Q : \left\| D_k^{-1}\left(\pi - |\pi|^k\right)\right\|_\infty \leqslant 1\right\}, \quad D_k = \mathrm{diag}\left(\delta_1^k, \ldots, \delta_T^k\right). \tag{4.21}$$

For $t \in \mathcal{T}$ and $k \geqslant 2$, let

$$s_t^k = \left|\pi_t^k - |\pi|_t^{k-1}\right|, \quad \delta_t^1 = +\infty, \quad \delta_t^2 = \left\|\pi^2 - \pi^1\right\|_\infty, \quad n_t^2 = 0, \tag{4.22}$$

and, for $k \geqslant 3$,

$$n_t^k = \begin{cases} 0 & \text{if } \ f(\pi^k) < UB_{k-1}, \\ \left(n_t^{k-1} \bmod 3\right) + 1 & \text{if } \ s_t^k \geqslant 0.9\delta_t^{k-1}, \ f(\pi^k) \geqslant UB_{k-1}, \\ n_t^{k-1} & \text{otherwise,} \end{cases} \tag{4.23}$$

$$\delta_t^k = \begin{cases} 2\delta_t^{k-1} & \text{if } \ s_t^k \geqslant 0.9\delta_t^{k-1} \ \text{ and } \ f(\pi^k) < UB_{k-1}, \\ \delta_t^{k-1}/2 & \text{if } \ s_t^k \geqslant 0.9\delta_t^{k-1}, \ f(\pi^k) \geqslant UB_{k-1} \ \text{ and } \ n_t^k = 3, \\ \delta_t^{k-1} & \text{otherwise.} \end{cases} \tag{4.24}$$

The radii follow the usual trust-region logic, applied to each period separately. A step that hits the boundary of $X_{k-1}$ in period $t$ and improves the best value suggests that the region is too tight in that period, so $\delta_t$ is doubled. Repeated boundary steps without improvement suggest the opposite, and $\delta_t$ is halved after three of them. Because $X_k$ is smaller than Q, the model minimum over $X_k$ is only a local bound, so it is recomputed at each iteration rather than accumulated:

$$LB_k = \max\left\{LB_0, \min_{\pi \in X_k} m_k(\pi)\right\}. \tag{4.25}$$

If this minimum is not below $UB_k$, the model predicts no decrease inside $X_k$, and we double all radii until it does. This lower bound is valid whenever $X_k$ contains a minimizer of f, which held at every iteration of our runs.

We write PC-BLM and PC-BPLM for the variants of BLM and BPLM that use this diagonal metric. These variants can be proven to converge asymptotically using the same argument as for Kelley's cutting-plane method, although we do not know whether a convergence rate can be established for them.

**Dynamic level parameter (DLM).** Zhang et al. (2024) propose adjusting the level parameter $\alpha$ dynamically rather than keeping it fixed throughout the run. The basic idea is to reduce $\alpha$, thereby tightening the level set, when the gap $\Delta = UB - LB$ decreases sufficiently, and to increase it otherwise. This is similar in spirit to a trust-region mechanism, in which the admissible region is enlarged or contracted depending on the observed progress.

Let $\alpha_k \in (0, 1)$ be the level parameter, initialized at a tuned $\alpha_0$. Having computed $\pi^{k+1}$, we form the ratio of achieved to targeted reduction in the upper bound,

$$\rho_k = \frac{UB_k - \min\{UB_k, f(\pi^{k+1})\}}{\max\{10^{-12}, UB_k - \ell_k\}}. \tag{4.26}$$

The level parameter is then updated by

$$\alpha_{k+1} = \begin{cases} \omega\alpha_k & \text{if } \rho_k > \rho_{up}, \\ 1 - \omega(1 - \alpha_k) & \text{if } 0 < \rho_k \leqslant \rho_{up}, \\ \alpha_k & \text{otherwise.} \end{cases} \tag{4.27}$$

Here $\rho_{up} = 0.5$ and $\omega = 0.5$. A smaller $\alpha_k$ moves the level value toward the lower bound and tightens the next level set.

We denote this variant by DLM and include it in our benchmark to assess adaptive level control in the CH pricing context.

Recent parameter-free first-order methods Parameter-free methods have regained significant attention during the last few years and multiple new methods have been proposed, in particular for the purpose of optimization for machine learning, see Nesterov (2015), Orabona & Pál (2016), Cutkosky & Boahen (2017), Roulet & d'Aspremont (2017), Defazio & Mishchenko (2023), Khaled et al. (2023), and Li & Lan (2023). We present two methods among the latter, D-Adaptation (Defazio & Mishchenko, 2023) and DoWG (Khaled et al., 2023), and adapt them to the CH pricing problem.

**Algorithm 4.6 (D-Adaptation (DA))**:

**Parameters:** initial distance estimate $D_0 > 0$
**Inputs:** initial iterate $\pi^1 \in Q$
Initialize $s^0 = 0$, $G_0 = 0$, **and** $A_0 = 0$.
**repeat for** $k = 1, 2, ...$
Select $g^k \in \partial f(\pi^k)$.
Set $s^k = s^{k-1} + D_{k-1} g^k$ **and** $G_k = G_{k-1} + \|g^k\|^2$.
Set $\gamma_k = \frac{1}{\sqrt{G_k}}$ **if** $G_k > 0$
Set $A_k = A_{k-1} + \gamma_k D_{k-1}^2 \|g^k\|^2$.
Set $D_k = \max\left\{D_{k-1}, \frac{\gamma_k \|s^k\|^2 - A_k}{\max\{2\|s^k\|, 10^{-12}\}}\right\}$.
Compute $\pi^{k+1} = P_Q(\pi^1 - \gamma_k s^k)$.
**until** a stopping criterion is satisfied.
**Output:** best iterate $\pi_{\arg\min_k\{f(\pi_k)\}}$

**Algorithm 4.7 (Distance over Weighted Gradients (DoWG))**:

**Parameters:** initial distance estimate $d_0 > 0$
**Inputs:** initial iterate $\pi^1 \in Q$
Initialize $r_0 = d_0$ **and** $v_0 = 0$.
**repeat for** $k = 1, 2, ...$
Select $g^k \in \partial f(\pi^k)$.
Set $r_k = \max\{r_{k-1}, \|\pi^k - \pi^1\|\}$.
Set $v_k = v_{k-1} + r_k^2 \|g^k\|^2$.
Set $\eta_k = \frac{r_k^2}{\sqrt{v_k}}$ **if** $v_k > 0$.
Compute $\pi^{k+1} = P_Q(\pi^k - \eta_k g^k)$.
**until** a stopping criterion is satisfied.
**Output:** best iterate $\pi_{\arg\min_k\{f(\pi_k)\}}$

Both methods try to circumvent the issue of not knowing the distance to the optimal solution $R$ by using an initial distance estimate. This estimate is then updated at each iteration to approximate the true distance. This approach helps select the stepsize adaptively and allows convergence without knowing any of the problem parameters.

### 4.3.3 Smoothing-based methods

First-order methods for convex nonsmooth minimization have a convergence rate of order $\mathcal{O}(\epsilon^{-2})$, where $\epsilon$ is the desired accuracy, and this order is optimal. On the other hand, simple convex smooth minimization methods have convergence rates of order $\mathcal{O}(\epsilon^{-1})$ or $\mathcal{O}(\epsilon^{-0.5})$ for the fastest algorithms. To bypass the slow nonsmooth rate, we use a smoothed version of the convex objective $f = -\mathcal{L}$ based on Nesterov's smoothing technique (Nesterov, 2005) and apply a fast projected gradient method (Beck & Teboulle, 2009).

We recall that $f = f_0 + \sum_{g\in\mathcal{G}} f_g$ with $f_0 = -\mathcal{L}_0$ and $f_g = -\mathcal{L}_g$, where $\mathcal{L}_0$ and $\mathcal{L}_g$ are given in (4.9) and (4.10). We smooth both $\mathcal{L}_0$ and $\mathcal{L}_g$, which are nondifferentiable and given as explicit minimizations. Given reference points $l^r$ and $(\mathbf{p}^g, \mathbf{u}^g, \boldsymbol{\nu}^g, \boldsymbol{w}^g)^r$ and a smoothing parameter $\varsigma > 0$, we introduce

$$\begin{aligned}\mathcal{L}_{0,\varsigma}(\pi) = \min_{l} \sum_{t\in\mathcal{T}} [C_{\mathrm{VOLL}}(L_t - l_t) + \pi_t l_t] + \frac{\varsigma}{2}\|l - l^r\|^2 \\ \text{s.t.} \quad 0 \leqslant l_t \leqslant L_t, \quad \forall t \in \mathcal{T},\end{aligned} \tag{4.28}$$

$$\begin{aligned}\mathcal{L}_{g,\varsigma}(\pi) = \min_{(\mathbf{p}^g,\mathbf{u}^g,\boldsymbol{\nu}^g,\boldsymbol{w}^g)} \sum_{t\in\mathcal{T}} \left[C(p_t^g, u_t^g, \nu_t^g) - \pi_t p_t^g\right] \\ + \frac{\varsigma}{2}\left\|(\mathbf{p}^g, \mathbf{u}^g, \boldsymbol{\nu}^g, \boldsymbol{w}^g) - (\mathbf{p}^g, \mathbf{u}^g, \boldsymbol{\nu}^g, \boldsymbol{w}^g)^r\right\|^2 \\ \text{s.t.} \quad (\mathbf{p}^g, \mathbf{u}^g, \boldsymbol{\nu}^g, \boldsymbol{w}^g) \in \mathrm{conv}(\mathcal{P}^g),\end{aligned} \tag{4.29}$$

the smooth approximations of $\mathcal{L}_0$ and $\mathcal{L}_g$. Since the objective of (4.10) is linear, replacing $\mathcal{P}^g$ by its convex hull does not change $\mathcal{L}_g$, and (4.29) becomes a strongly convex problem over a polytope. Note that adding regularization to the original problem (4.10), defined on the nonconvex set $\mathcal{P}^g$, would not lead to a strongly convex objective: passing to the convex hull of the feasible region is required. The smoothed objective is

$$\tilde{f}_\varsigma(\pi) = -\mathcal{L}_{0,\varsigma}(\pi) - \sum_{g\in\mathcal{G}} \mathcal{L}_{g,\varsigma}(\pi). \tag{4.30}$$

**Lemma 4.4.** *The function $\tilde{f}_\varsigma$ is convex and differentiable. For any $\pi$, let $l^*_\varsigma(\pi)$ be the unique solution of (4.28) and $p^g_{\varsigma,*}(\pi)$ the production part of the unique solution of (4.29). Then*

$$\nabla \tilde{f}_\varsigma(\pi) = \sum_{g\in\mathcal{G}} p^g_{\varsigma,*}(\pi) - l^*_\varsigma(\pi), \tag{4.31}$$

*and $\nabla\tilde{f}_\varsigma$ is $(1 + |\mathcal{G}|)/\varsigma$-Lipschitz continuous.*

*Proof. Gradient.* Fix $g \in \mathcal{G}$, write $x = (p^g, u^g, v^g, w^g)$ and let $q^g_\pi(x)$ be the objective of (4.29). Since $q^g_\pi$ is $\varsigma$-strongly convex and $\operatorname{conv}(\mathcal{P}^g)$ is a nonempty compact convex set, (4.29) has a unique solution $x^g_*(\pi)$. The function $-\mathcal{L}_{g,\varsigma}$ is a pointwise maximum of affine functions of $\pi$, hence convex, and since the maximizer is unique, Danskin's theorem shows that it is differentiable with gradient $p^g_{\varsigma,*}(\pi)$. The same argument applied to (4.28) shows that $-\mathcal{L}_{0,\varsigma}$ is convex and differentiable with gradient $-l^*_\varsigma(\pi)$, and summing over the components of (4.30) gives the expression of $\nabla \tilde{f}_\varsigma$.

*Lipschitz continuity.* Let $\pi_1, \pi_2 \in \mathbb{R}^{|\mathcal{T}|}$, $x_i = x^g_*(\pi_i)$ and $p_i$ its production part, for $i = 1, 2$. By $\varsigma$-strong convexity of $q^g_{\pi_2}$ and optimality of $x_2$ over $\operatorname{conv}(\mathcal{P}^g)$, which gives $\langle \nabla q^g_{\pi_2}(x_2), x_1 - x_2 \rangle \geqslant 0$, we have

$$\begin{aligned} \frac{\varsigma}{2}\|x_1 - x_2\|^2 &\leqslant q^g_{\pi_2}(x_1) - q^g_{\pi_2}(x_2) - \langle \nabla q^g_{\pi_2}(x_2), x_1 - x_2 \rangle \\ &\leqslant q^g_{\pi_2}(x_1) - q^g_{\pi_2}(x_2), \end{aligned} \tag{4.32}$$

and, exchanging the roles of $\pi_1$ and $\pi_2$,

$$\frac{\varsigma}{2}\|x_1 - x_2\|^2 \leqslant q^g_{\pi_1}(x_2) - q^g_{\pi_1}(x_1). \tag{4.33}$$

Since $q^g_{\pi_1}(x) - q^g_{\pi_2}(x) = -\langle \pi_1 - \pi_2, p \rangle$ for any schedule $x$ with production part $p$, summing the two inequalities gives

$$\varsigma\|x_1 - x_2\|^2 \leqslant \langle \pi_1 - \pi_2, p_1 - p_2 \rangle. \tag{4.34}$$

By the Cauchy–Schwarz inequality and $\|p_1 - p_2\| \leqslant \|x_1 - x_2\|$, we obtain

$$\|p_1 - p_2\| \leqslant \|x_1 - x_2\| \leqslant \frac{1}{\varsigma}\|\pi_1 - \pi_2\|. \tag{4.35}$$

The same bound holds for $l^*_\varsigma$, and summing over the $1 + |\mathcal{G}|$ components of (4.30) gives the Lipschitz constant $(1 + |\mathcal{G}|)/\varsigma$. □

By the lemma, the gradient of $\tilde{f}_\varsigma$ requires the production part of the solution of (4.29) for each generator. We compute this solution by column generation on the vertex representation of $\operatorname{conv}(\mathcal{P}^g)$. Given a set $V^g \subset \mathcal{P}^g$ of schedules, the restricted master problem $\min_{x \in \operatorname{conv}(V^g)} q^g_\pi(x)$ is a convex quadratic program in the weights of the schedules. Its solution $\bar{x}^g$ gives the costs $d^g = \nabla q^g_\pi(\bar{x}^g)$, and the pricing problem $\min_{x \in \mathcal{P}^g} (d^g)^\top x$ is (4.10) with costs $d^g$, a MILP of the same size as the subproblems of the other methods. Let $B^g$ be a lower bound on its value, such as the best bound returned by the MILP solver, and $\delta^g = (d^g)^\top \bar{x}^g - B^g$. We stop when

$\delta^g \leqslant \tau$ or after 50 master iterations, and otherwise add the pricing solution to $V^g$ (Figure 4.2). The columns are kept from one FGM iteration to the next: consecutive prices are close, so the previous columns already describe most of the new solution and few additional master iterations are needed. The generators are handled in parallel, and only MILPs and convex quadratic programs are solved.

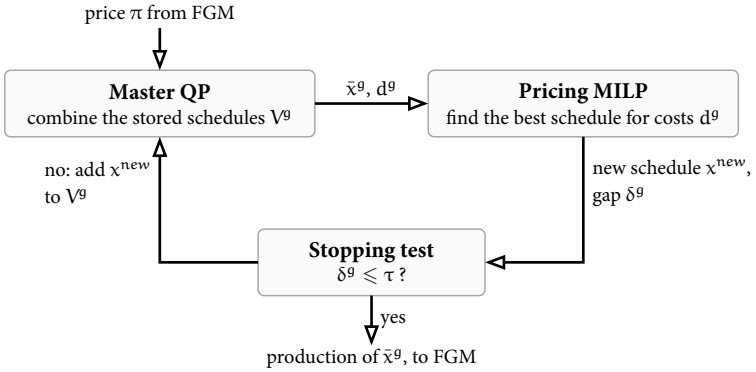



Figure 4.2: Column generation for (4.29) at the price $\pi$. The master returns the best point $\bar{x}^g$ in the convex hull of the stored schedules $V^g$. The pricing MILP minimizes over $\mathcal{P}^g$ the linearization of $q_\pi^g$ at $\bar{x}^g$, with costs $d^g = \nabla q_\pi^g(\bar{x}^g)$, and returns a new schedule and the gap $\delta^g = (d^g)^\top \bar{x}^g - B^g$.

**Proposition 4.5.** *Let $\bar{x}^g$ and $\delta^g$ be returned by the column generation, $x_*^g$ the solution of (4.29), and $\bar{p}^g$, $p_*^g$ their production parts. Then*

$$0 \leqslant q_\pi^g(\bar{x}^g) - \mathcal{L}_{g,\varsigma}(\pi) \leqslant \delta^g \quad \text{and} \quad \|\bar{p}^g - p_*^g\| \leqslant \sqrt{2\delta^g/\varsigma}. \tag{4.36}$$

*Proof.* Since $\bar{x}^g \in \mathrm{conv}(\mathcal{P}^g)$, we have $\mathcal{L}_{g,\varsigma}(\pi) = q_\pi^g(x_*^g) \leqslant q_\pi^g(\bar{x}^g)$. Moreover, a linear function attains its minimum over the polytope $\mathrm{conv}(\mathcal{P}^g)$ at a vertex, and the vertices of $\mathrm{conv}(\mathcal{P}^g)$ belong to $\mathcal{P}^g$. Hence, since $B^g$ is a lower bound on the value of the pricing problem,

$$\langle d^g, x_*^g \rangle \geqslant \min_{x \in \mathcal{P}^g} \langle d^g, x \rangle \geqslant B^g. \tag{4.37}$$

By convexity of $q_\pi^g$ and since $d^g = \nabla q_\pi^g(\bar{x}^g)$, we have

$$\mathcal{L}_{g,\varsigma}(\pi) = q_\pi^g(x_*^g) \geqslant q_\pi^g(\bar{x}^g) + \langle d^g, x_*^g - \bar{x}^g \rangle \geqslant q_\pi^g(\bar{x}^g) - \delta^g, \tag{4.38}$$

which proves the first claim. By $\varsigma$-strong convexity of $q_\pi^g$ and optimality of $x_*^g$ over $\mathrm{conv}(\mathcal{P}^g)$, which gives $\langle \nabla q_\pi^g(x_*^g), \bar{x}^g - x_*^g \rangle \geqslant 0$, we have

$$\frac{\varsigma}{2}\|\bar{x}^g - x_*^g\|^2 \leqslant q_\pi^g(\bar{x}^g) - q_\pi^g(x_*^g) - \langle \nabla q_\pi^g(x_*^g), \bar{x}^g - x_*^g \rangle \leqslant \delta^g, \qquad (4.39)$$

where the last inequality uses the first claim. Since $\|\bar{p}^g - p_*^g\| \leqslant \|\bar{x}^g - x_*^g\|$, this gives the second claim. □

The bound only uses $B^g$, so it also holds when the pricing MILPs are not solved to optimality. At each FGM iteration, the subproblems are thus solved within $\epsilon = \sum_{g\in\mathcal{G}} \delta^g$, which makes the oracle a $(2\epsilon, 2L)$-oracle with $L = (1 + |\mathcal{G}|)/\varsigma$ (Devolder et al., 2014, Section 3.1). Following (Devolder et al., 2014, Equation 39), FGM with this inexact oracle converges on the smoothed objective at the rate $\mathcal{O}(1/k^2)$, up to an additional accumulated error $2(k+3)\epsilon/3$ that grows linearly with $k$. With the tolerance $\tau = 10^{-7}$ and the number of iterations FGM performs in 15 minutes, this bound amounts to a few cents over a whole market, far below target accuracies encountered in practice.

We take as fixed reference points the optimal solutions of the continuous linear relaxation of Problem 4.1. The fast gradient method used to minimize the smoothed objective is given in Algorithm 4.8.

**Algorithm 4.8 (Fast (Projected) Gradient Method (FGM))**:

**Parameters:** step size $\eta$, smoothing parameter $\varsigma$
**Inputs:** initial iterate $\pi^1 \in Q$
Initialize $y^1 = \pi^1$.
**repeat for** $k = 1, 2, \ldots$
Compute $\nabla \tilde{f}_\varsigma(y^k)$.
Compute $\pi^{k+1} = P_Q(y^k - \eta \nabla \tilde{f}_\varsigma(y^k))$.
Compute $y^{k+1} = \pi^{k+1} + \frac{k-1}{k+2}(\pi^{k+1} - \pi^k)$.
**until** a stopping criterion is satisfied.
Set $k_{best} \in \arg\min_{1\leqslant i\leqslant k} f(\pi^i)$ **and** output $\pi^{k_{best}}$.

In the following, the values reported for FGM are computed on the original objective $f(\pi)$ evaluated at the iterates generated by the smoothed problem, not on the surrogate $\tilde{f}_\varsigma(\pi)$. This makes the reported objective values directly comparable with those of the other methods.

Let $D$ denote the prox-diameter of the primal feasible region. The discrepancy between the two objectives is bounded by the standard smoothing estimate, which holds for any $\pi$:

$$0 \leqslant f(\pi) - \tilde{f}_\varsigma(\pi) \leqslant \varsigma D, \tag{4.40}$$

see (Nesterov, 2005, Equation 2.7). Consequently, any minimizer of $\tilde{f}_\varsigma$ over $Q$ is $\varsigma D$ -optimal for $f$, so that as $\varsigma \to 0$, every limit point of these minimizers is a vector of CH prices.

The reference point used in the smoothing construction acts as a primal selection mechanism: for a given $\pi$, it favors primal minimizers close to the reference point when multiple minimizers exist. It therefore affects the gradient of the smoothed dual and the geometry of the surrogate problem solved by FGM, but it does not modify the original CH-pricing dual objective on which the final prices are evaluated.

## 4.4 Practical implementation

The `ConvexHullPricing.jl`[6] package implements primal and dual optimization methods for computing convex hull prices. It uses a common model to represent thermal generators and unit-commitment instances. The package is mainly designed to benchmark optimization methods for this problem, and it also computes settlement quantities that are useful to practitioners.

**Design principles.** The structure of the implementation is summarized in Figure 4.3.

[6] https://github.com/SofianeTanji/PhD-Software/ConvexHullPricing.jl

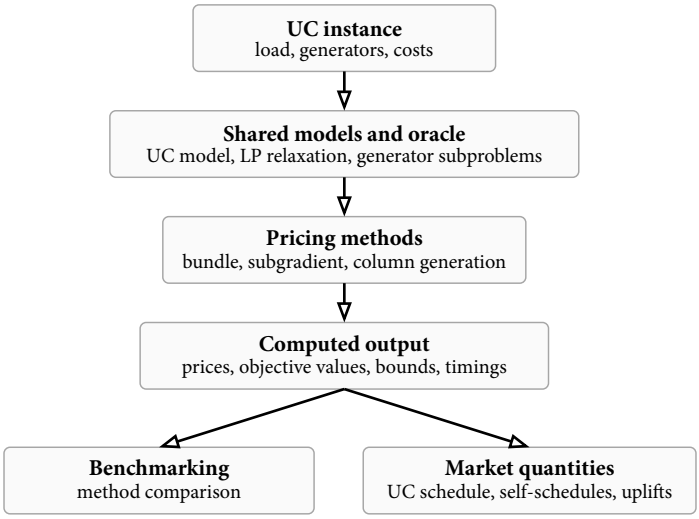



Figure 4.3: Main workflow in the `ConvexHullPricing.jl` package

We keep the instance, the pricing methods, and the settlement calculations separate, so that each can be varied without touching the others. A unit-commitment instance is given by its load, its generators, and their cost functions. The pricing methods reach it only through the oracle, which returns the dual objective and its subgradient at a given price vector. The methods can then be benchmarked against one another, as in Section 4.5, and the computed prices yield the market quantities of interest to practitioners, among which: the market schedule, the self-schedules, and the uplifts.

**Computing prices.** After defining a unit-commitment instance, the package can compute convex hull prices using any of the methods described in Section 4.3. The linear relaxation gives the initial price vector, and the chosen method then runs under a fixed budget. All methods record at least the computed prices, the successive objective values, and the elapsed times, which are useful for benchmarking. For the variable-metric BLM/BPLM variants, we fix the number of clusters to $K = 50$ before the run.

**Reusing models across iterations.** The JuMP models of the generator subproblems are built once and kept in memory for the whole run[7]. At each oracle call, only the objective changes, to reflect the current prices. Similarly, the master problems

[7] To further improve the efficiency of the implementation, we create these models with JuMP's `direct_model`, which keeps no intermediate copy of the model and passes every modification directly to the underlying Gurobi model.

of both the bundle methods and column generation are persistent: at each iteration, we only add cuts or columns to the model already in memory.

**From prices to market quantities.** The package also connects the computed prices to the market quantities that make them interpretable. The market schedule is obtained by solving the centralized binary unit-commitment problem. For a fixed price vector, it solves each generator's self-scheduling problem and compares the resulting profit with the profit assigned by the market schedule. The positive part of this difference is the generator's lost opportunity cost. Aggregating these quantities gives the total uplift, together with per-generator production, cost, revenue, and profit.

The complete workflow can be summarized by the following short script:

```
using ConvexHullPricing

instance = load_data("case.json")
budget = 300.0
prices = compute_ch_prices(instance; budget)
market_schedule = solve_market_schedule(instance)
uplift = total_uplift(instance, prices, market_schedule)
```

## 4.5 Numerical experiments

We now study the performance of the proposed methods on real-world UC instances. We first describe the common experimental protocol and the implementation choices shared by all methods. We then describe the main implementation choices for the multi-cut bundle variants and compare the bundle methods within this family to determine which variants are retained for the main benchmark. We next compare these retained bundle methods with the other algorithmic families. Finally, we report diagnostic and robustness analyses.

### 4.5.1 Experimental protocol

**Experimental setup.** Each method is implemented in Julia (1.12.5) using JuMP (1.29.4) and Gurobi (13.0). Oracle subproblems use Gurobi default parameters except for the relative MIP gap tolerance, which is set to $10^{-8}$. All computations use the implementation described in Section 4.4. All tuning and benchmark runs were performed on AMD EPYC 9655 processors at 2.6 GHz, using 8 threads and 15 GB of RAM per run.

**Instance sets.** We run all experiments on 24 real-world UC problem instances. The *Belgian* instance set contains 8 real-world instances, one for each season and

weekday/weekend combination, with 68 Belgian generators over a 24-hour horizon with 15-minute time periods. The *Californian* instance set contains 16 curated instances[8] extracted from publicly available California ISO data[9].

Considering instances where the price space has dimensions 48 and 96 (i.e., one price per period over a 24-hour horizon) allows us to examine how sensitive the performance results are to the dimensionality of the price space. Instances of this type, without a transmission network, were analyzed in Knueven et al. (2022) and Stevens & Papavasiliou (2022). The role of problem dimensionality was further emphasized in Stevens & Papavasiliou (2022), which additionally considered instances with multiple locations connected through a transmission network to investigate its impact in greater detail. Even without multiple locations, they observe that the standard Dantzig-Wolfe approach tested in Andrianesis et al. (2021) is already adversely impacted by an increase from 12 to 96 periods, in contrast to the Level method. In this work, we focus on similar instance types with a single location but while still having such a high number of periods, common to references Knueven et al. (2022), Stevens & Papavasiliou (2022), and Andrianesis et al. (2021), both to focus on the breadth of the dual methods tested and to facilitate future comparisons with the primal methods discussed in Andrianesis et al. (2021) and Knueven et al. (2022). Future work could explore extending the numerical experiments to instances with multiple locations connected by advanced transmission networks. We summarize these instance sets in the table below:

Table 4.1: Summary of the instance sets.

| Instance set | # instances | # generators | # time periods | source |
|---|---|---|---|---|
| Belgian | 8 | 68 | 96 | (Paquet, 2021; Stevens & Papavasiliou, 2022) |
| Californian | 16 | 610 | 48 | (Knueven et al., 2020) |

The 48 time periods in the Californian instance set correspond to half-hour intervals over a 24-hour horizon, while the Belgian instance set uses 96 intervals of 15 minutes.

Computation of reference values f* All considered instance sets originally describe unit commitment instances and therefore do not include reference solutions

[8] Among the 20 instances in the provided repository, we exclude 4 of them because they are not relevant in comparing methods, as our warm-start technique directly provides CH prices.

[9] Data available at https://github.com/power-grid-lib/pglib-uc.

to the CH pricing problem. Because CH prices may not be unique, we compare methods by (relative) objective accuracy rather than distance to solution set (which is unavailable). For each instance, we compute a high-accuracy reference minimum $f^*$ of the convex dual objective. On each of the 24 instances, we run BLM with parameter $\alpha = 0.2$ and Gurobi configured to output exact oracles (we set absolute and relative MIP gaps to zero) until the absolute gap between the constructed upper and lower bounds is below 1 unit of local currency. These computations took up to 6 hours. The reference value $f^*$ is not provided to the tested methods and is only used for comparison purposes.

**Hyperparameter selection.** For each method, we tune one main exposed parameter. Hence, the tuning procedure does not unfairly benefit any approach. Benchmarks in the next subsection are computed after tuning all methods on one instance per dataset (on *Autumn Week-day* for the *Belgian* dataset and on *2014-09-01_reserves_0* for the *Californian* dataset). To perform our hyperparameter tuning, we run each method for 5 minutes for each value of the hyperparameter belonging to a coarse grid with 6 values, logarithmic except for the level parameters of the bundle methods, for which it is linear. We then make a second finer grid search with 4 values around the best found parameter in the coarse grid search. Results of the hyperparameter tuning are found in the table below:

Table 4.2: List of tuned hyperparameters for each considered optimization method

| Method | Hyperparameter | BE | CA |
|---|---|---|---|
| **Subgradient methods** | | | |
| SUBG | Initial step length $\eta \in [10^{-2}; 10^{3}]$ | 290 | 0.40 |
| SUBG-EP | Initial error estimate $\alpha \in [10^{1}; 10^{5}]$ | 63000 | 250 |
| SUBG-L | Initial distance estimate $R \in [10^{-2}; 10^{3}]$ | 160 | 0.40 |
| **Bundle methods** | | | |
| BLM | Level parameter $\alpha \in [0.1, 0.99]$ | 0.99 | 0.99 |
| PC-BLM | Level parameter $\alpha \in [0.1, 0.99]$ | 0.75 | 0.63 |
| PC-BPLM | Level parameter $\alpha \in [0.1, 0.99]$ | 0.81 | 0.40 |
| DLM | Initial level parameter $\alpha_0 \in [0.1, 0.99]$ | 0.93 | 0.99 |
| **Parameter-free methods** | | | |
| DA | Initial distance estimate $D_1 \in [10^{-2}; 10^{2}]$ | 16 | 0.1 |
| DOWG | Initial distance estimate $d_0 \in [10^{-2}; 10^{2}]$ | 10 | 0.025 |
| **Smoothing-based method** | | | |
| FGM | Constant step length $\eta \in [10^{-7}; 10^{-2}]$ | $2.2 \times 10^{-5}$ | $4.0 \times 10^{-6}$ |
| **Primal method** | | | |
| CG | Tolerance parameter $\epsilon \in [10^{-9}, 10^{1}]$ | $10^{-6}$ | $10^{-6}$ |

The *SUBG-L* method also requires the total number of iterations as an input parameter. To align its run time with other methods, we pick the number of iterations that results in a total computation time of about 15 minutes. *FGM* works on a smoothed variant of the original instance; its smoothing parameter is fixed at $10^{-8}$ for both instance sets.

Warm starts For all methods, we set the first iterate to be the dual variables of the power-balance constraints (4.4) in the continuous relaxation of the UC problem. This initialization has low cost and reduces the order of magnitude of the Lagrangian function compared with fixed-price initializations such as $\pi_1 = [40, 40, ..., 40]$. This initialization is interpretable: these prices are exact convex hull prices when only minimum up and down times and constant startup or shutdown costs are present and the classic tight 3-bin formulation is used; see Hua & Baldick (2016).

### 4.5.2 Stabilized implementation of multicut bundle variants

In preliminary experiments, the direct multicut formulation was numerically unstable. The main difficulty arises in the projection problem: each iteration adds one cut per generator, so after a few iterations the master problem contains many nearly redundant linear inequalities. This degrades conditioning, especially because nonunique mixed-integer optima can generate different valid subgradients at the same or nearby price vectors. PC-BLM and PC-BPLM rely on this stabilized implementation, while MC-BLM and MC-BPLM keep one cut per generator.

The stabilization addresses this issue by controlling the growth and redundancy of the cut system. A first possibility is to aggregate or prune cuts across iterations, but this sacrifices the monotonic decrease of the lower bound. We instead adopt a clustered multi-cut construction. At each iteration, the generator-wise cuts are computed as usual, but generators are partitioned once into $K$ groups, according to the time period at which the absolute value of the subgradient is maximal, which serves as a computationally inexpensive proxy for clustering cuts with similar orientation. The master problem then uses one epigraph variable and one cut per cluster rather than per generator. This preserves the multi-cut structure while keeping the projection problem numerically manageable.

### 4.5.3 Selection of bundle variants

We compare several bundle variants to decide which methods to retain for the main benchmark. We first set aside Kelley's cutting-plane method, which uses the same cutting-plane model and price box as BLM but moves to a minimizer of the model instead of projecting onto a level set. It does not reach the target used below, an excess side payment of $10^{-5}$ times the instance set's mean minimum side payment, within 15 minutes on any of the 8 Belgian instances and on 5 of the 16 Californian instances. To isolate the effect of the main modeling choices, we compare the single-cut baseline BLM with five variants: DLM, the multicut methods MC-BLM and MC-BPLM, and the variable-metric BLM/BPLM variants PC-BLM and PC-BPLM.

Table 4.3: Geometric-mean speedup of bundle variants relative to BLM. The target is an excess side payment of $10^{-5}$ times the instance set's mean minimum side payment.

| Set | BLM | DLM | MC-BLM | MC-BPLM | PC-BLM | PC-BPLM |
|---|---|---|---|---|---|---|
| BE | 1.0 × | 0.31 × | 0.99 × | 1.1 × | 2.4 × | **2.4** × |
| CA | 1.0 × | 0.75 × | 1.2 × | 1.3 × | **2.9** × | 2.5 × |

Table 4.3 shows that the generator-wise multicut model alone barely improves on BLM, and that most of the improvement comes from PC-BLM and PC-BPLM. PC-BLM and PC-BPLM give the largest speedups. On the Californian set, PC-BLM and PC-BPLM give speedups of 2.9 $\times$ and 2.5 $\times$. We retain PC-BLM and PC-BPLM for the main benchmark below. The threshold here is based on the instance set's mean minimum side payment, while the main benchmark uses an instance-specific reference optimum. Speedups under the two protocols are therefore not directly comparable.

### 4.5.4 Main benchmark

We now compare the retained bundle methods, PC-BLM and PC-BPLM, with the subgradient methods, adaptive first-order methods, the smoothing-based method, and a primal column-generation reference, which we denote by CG. By column generation we mean the Dantzig-Wolfe decomposition of the primal UC problem discussed in Section 4.1, which amounts to Kelley's cutting-plane algorithm on the dual. We implement it with a multicut polyhedral master problem for fair comparison with the bundle methods, and include it here as the primal counterpart of the dual methods that are our subject. All methods are evaluated on the 24 instances described above with a time limit of 15 minutes. The results are reported in Figure 4.4. For each method, the plots show the evolution over time of the mean objective error over all instances in each instance set.

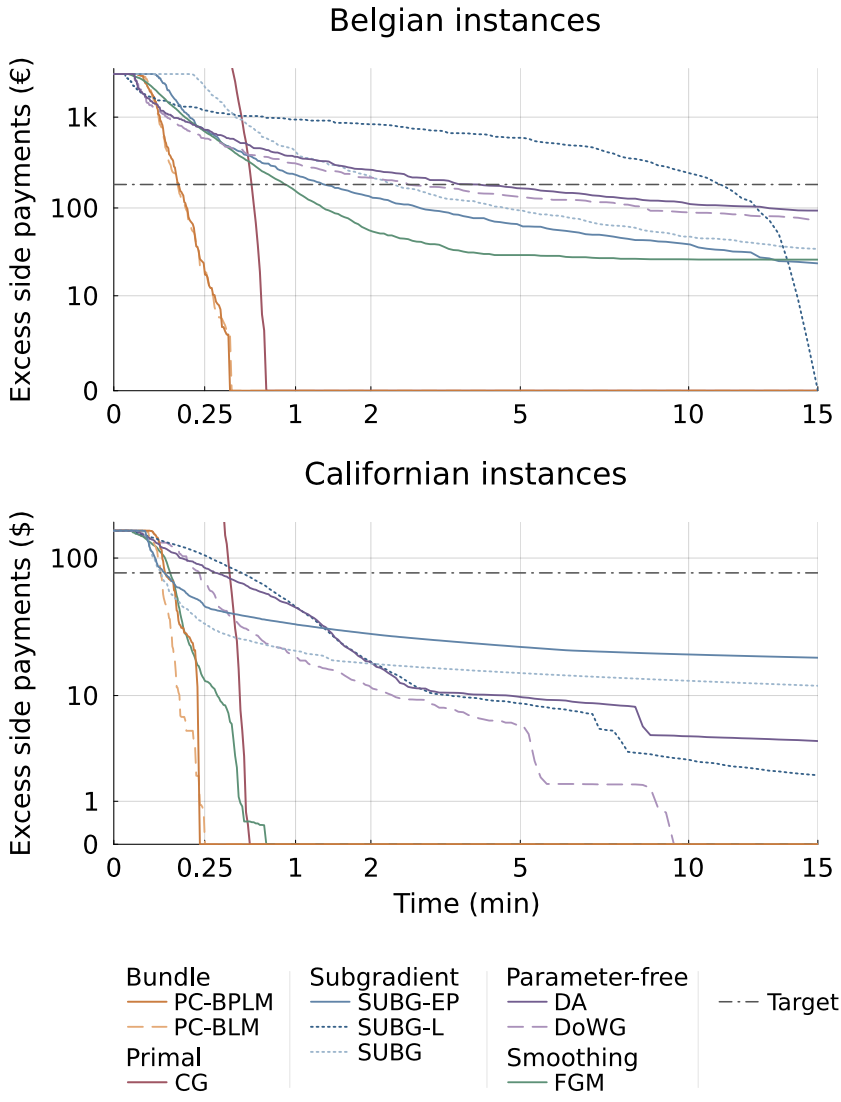



Figure 4.4: Mean excess side payments over time. Curves are evaluated on a common time grid and averaged arithmetically over instances. Wall time is shown on a logarithmic scale. The dash-dot line marks the target $10^{-5}$ times the mean minimum side payment. The corresponding mean minimum side payments are 18.17 million euros for Belgium and 7.85 million dollars for California.

The main findings are as follows.

- **Multicut polyhedral methods reach the specified numerical tolerance fastest on the Belgian instances.** All methods reach the objective-error tolerance used in Table 4.5 on all instances within 15 minutes, but on the Belgian instances PC-BLM and PC-BPLM do so at least five times faster than every other method.
- **Bundle methods give stronger intermediate solutions than CG.** CG starts without enough columns to cover the demand, so its restricted master problem sheds load at the value of lost load and its primal value remains far above the optimum during the first iterations. CG reaches the target line after about 0.6 minutes on the Belgian instance set and 0.4 minutes on the Californian instance set, against about 0.1 minutes for the retained bundle methods, which start from the continuous-relaxation prices and decrease excess side payments steadily.
- **Subgradient methods give strong early solutions on the Californian instances.** SUBG-EP reaches the target line in about 0.08 minutes on the Californian instance set, as fast as the retained bundle methods, and SUBG-L in about 0.50 minutes, while the retained bundle methods are stronger on the Belgian instance set.
- **FGM is the fastest method outside the bundle family on the Belgian instances.** On the Belgian instance set, FGM reaches the target line after about 0.9 minutes, before every subgradient and parameter-free method, and its geometric-mean time to reach the tolerance of Table 4.5 is 36.3 seconds, against 6.7 seconds for the retained bundle methods and at least 79.6 seconds for the other methods. On the Californian instance set, it reaches the target line after about 0.1 minutes and has the smallest geometric-mean time to reach this tolerance, 3.5 seconds.
- **The adaptive first-order methods are secondary in this benchmark.** On the Belgian instance set, DA and DoWG end around €94 and €73, against €26 for FGM, whereas the retained bundle methods reach the target. On the Californian instance set, DA, DoWG, and FGM end within a few dollars of the optimum.

The proposed hyperparameter selection strategy uses a single tuning pass per dataset: one for the eight Belgian instances and one for the 16 Californian instances. To assess how this strategy affects performance, Table 4.4 reports the final excess side payments for all instances.

Table 4.4: Final objective errors after 15 minutes, in local currency units. Tuning instances are italicized.

| Set | Bundle | | Subgradient | | | Parameter-free Smoothing | | |
|---|---|---|---|---|---|---|---|---|
| | PC-BPLM | PC-BLM | SUBG-EP | SUBG-L | SUBG | DA | DoWG | FGM |
| BE | ***0.0*** | *0.1* | *100.9* | *5.9* | *81.2* | *198.9* | *177.5* | *24.0* |
| | **0.0** | 0.1 | 20.2 | 4.1 | 31.3 | 76.7 | 58.1 | 25.7 |
| | **0.0** | **0.0** | 14.5 | 1.5 | 29.3 | 79.7 | 47.6 | 25.3 |
| | **0.0** | 0.2 | 11.4 | 4.5 | 18.8 | 67.4 | 46.5 | 20.4 |
| | **0.0** | **0.0** | 19.3 | 3.5 | 25.8 | 87.9 | 55.2 | 25.5 |
| | 0.2 | **0.0** | 24.9 | 7.8 | 34.8 | 68.5 | 68.2 | 19.9 |
| | **0.0** | 0.2 | 21.8 | 4.9 | 29.6 | 93.1 | 74.0 | 26.1 |
| | **0.0** | 0.6 | 23.5 | 7.0 | 24.6 | 72.2 | 53.2 | 40.6 |
| CA | ***0.0*** | ***0.0*** | *0.3* | *0.6* | *0.5* | *1.2* | *2.7* | *4.2* |
| | **0.0** | 0.1 | 0.2 | 0.1 | 0.3 | 1.0 | 2.5 | 4.1 |
| | **0.0** | **0.0** | 0.4 | 0.2 | 0.4 | 1.1 | 2.5 | 3.9 |
| | **0.0** | 0.1 | 0.4 | 1.0 | 0.5 | 1.3 | 2.7 | 4.3 |
| | **0.0** | **0.0** | 18.7 | 8.1 | 20.7 | 16.7 | 3.1 | 3.1 |
| | 0.1 | **0.0** | 18.1 | 7.0 | 20.8 | 16.7 | 3.1 | 2.8 |
| | 0.1 | **0.0** | 18.7 | 8.9 | 20.7 | 16.9 | 3.4 | 3.1 |
| | 0.2 | **0.0** | 18.5 | 7.6 | 20.7 | 17.1 | 3.4 | 3.2 |
| | **0.0** | 0.3 | 0.5 | 0.4 | 3.7 | 0.4 | 0.6 | 3.2 |
| | **0.0** | **0.0** | 0.2 | 0.1 | 3.4 | 0.2 | 0.4 | 3.4 |
| | **0.0** | 0.1 | 0.2 | 0.2 | 3.6 | 0.2 | 0.5 | 2.8 |
| | **0.0** | **0.0** | 0.2 | 0.2 | 3.5 | 0.2 | 0.7 | 3.4 |
| | **0.0** | 0.3 | 57.8 | 6.0 | 26.1 | 6.6 | 7.1 | 10.0 |
| | **0.0** | 0.2 | 57.6 | 7.2 | 26.5 | 6.7 | 7.1 | 8.5 |
| | **0.0** | **0.0** | 58.2 | 6.8 | 26.3 | 7.0 | 7.4 | 7.3 |
| | **0.0** | 0.2 | 58.1 | 6.1 | 25.9 | 6.8 | 7.7 | 9.1 |

These results suggest that a single hyperparameter selection per instance set is adequate in practice. Indeed, for a given instance set and method, the final excess side

payments are generally comparable across instances, which indicates a reasonable degree of robustness of the selected hyperparameters.

Table 4.5: Time in seconds to reach an excess side payment of $10^{-5}$ times the reference minimum side payment.

| Set | Bundle | | Subgradient | | | Parameter-free | | Smoothing |
|---|---|---|---|---|---|---|---|---|
| | PC-BPLM | PC-BLM | SUBG-EP | SUBG-L | SUBG | DA | DoWG | FGM |
| BE | *12.2* | ***12.1*** | *171.6* | *688.9* | *236.4* | *595.9* | *508.6* | *110.6* |
| | **5.9** | 6.7 | 103.8 | 680.8 | 168.1 | 245.1 | 200.8 | 57.1 |
| | 8.0 | **7.2** | 52.2 | 611.5 | 121.0 | 104.3 | 57.0 | 18.7 |
| | **5.0** | 7.0 | 76.5 | 702.9 | 143.8 | 156.7 | 153.2 | 25.6 |
| | **4.3** | 5.5 | 63.2 | 702.9 | 116.8 | 201.9 | 90.8 | 23.6 |
| | 7.6 | **6.3** | 106.3 | 730.3 | 170.2 | 344.0 | 314.3 | 57.4 |
| | 6.4 | **6.1** | 67.5 | 568.2 | 108.6 | 154.2 | 79.1 | 31.8 |
| | 6.4 | **4.7** | 49.9 | 619.7 | 96.9 | 124.1 | 69.5 | 23.3 |
| CA | *3.3* | *2.5* | ***2.0*** | *11.2* | *2.4* | *5.2* | *5.8* | *3.5* |
| | 3.6 | 3.9 | 2.1 | 8.7 | **2.0** | 5.5 | 5.9 | 3.8 |
| | 6.5 | 3.0 | **2.1** | 8.3 | 2.6 | 5.1 | 5.5 | 3.3 |
| | 2.9 | 3.5 | **2.2** | 11.0 | 2.6 | 5.6 | 6.0 | 3.7 |
| | 6.4 | 4.3 | 2.9 | 4.8 | 4.4 | 4.8 | 5.0 | **2.3** |
| | 9.0 | 5.4 | 2.9 | 3.9 | 4.5 | 4.9 | 5.1 | **1.9** |
| | 11.5 | 12.2 | 3.0 | 5.1 | 4.5 | 5.1 | 5.5 | **2.3** |
| | 4.5 | 3.3 | 3.2 | 3.8 | 4.5 | 5.2 | 5.3 | **2.0** |
| | 6.5 | 5.3 | 2.4 | 5.1 | 2.7 | **2.3** | 3.1 | **2.3** |
| | 3.8 | 2.6 | **2.1** | 4.3 | 2.3 | 2.5 | 3.1 | 2.3 |
| | 2.7 | **2.3** | **2.3** | 4.2 | 2.7 | 2.4 | 2.9 | 2.6 |
| | 3.7 | 2.4 | **2.0** | 5.7 | 3.0 | 2.6 | 3.2 | 2.4 |
| | **4.5** | 6.8 | 33.0 | 71.6 | 6.9 | 70.2 | 18.9 | 8.5 |
| | 4.2 | **3.8** | 32.7 | 65.1 | 6.7 | 68.5 | 19.2 | 8.0 |
| | 7.1 | **4.4** | 34.4 | 74.0 | 7.3 | 73.3 | 19.3 | 8.0 |
| | 13.0 | **2.5** | 34.6 | 66.1 | 7.0 | 70.0 | 19.9 | 7.4 |

Table 4.5 shows the time each method required to reach the instance-specific threshold, an excess side payment of $10^{-5}$ times the reference minimum side payment. In the Belgian instance set, all displayed methods reached the threshold on all instances, and PC-BPLM and PC-BLM were the fastest, with geometric means of 6.7 s each, against 36.3 s for FGM. In the Californian instance set, all displayed methods reached the threshold on all instances, with geometric means of 5.2 s for PC-BPLM, 4.6 s for SUBG-EP, and 3.7 s for SUBG.

**Bundle method stopping criteria.** Bundle methods maintain an explicit gap $\Delta = \mathrm{UB} - \mathrm{LB}$ between a lower and an upper bound on $f_*$, so they can stop without knowing $f_*$. The threshold of Table 4.5 relies on the reference optimum, while $\Delta$ is available at every iteration. Each bundle method is stopped once $\Delta$ falls to $10^{-5}$ times the reference minimum side payment, the level used in Table 4.5. PC-BLM reaches this threshold on every instance, with mean crossing times of 9.3 s on BE and 5.7 s on CA. PC-BPLM reaches it with mean crossing times of 9.6 s on BE and 4.7 s on CA. In these runs, the objective error itself falls below this threshold after 6.7 s and 6.8 s on average on BE, and after 5.2 s and 4.6 s on CA, for PC-BLM and PC-BPLM respectively. Stopping when $\Delta$ falls below the threshold, rather than when the objective error does, therefore delays termination by about 40% on the Belgian instances and only slightly on the Californian ones. When $\Delta$ reaches the threshold on the Belgian instances, the objective error is between $2.3 \cdot 10^{-6}$ and $4.4 \cdot 10^{-6}$ times the reference minimum side payment, and $\Delta$ exceeds it by a factor between 1.7 and 4.2. Since $\Delta$ is the sum of $\mathrm{UB} - f_*$ and $f_* - \mathrm{LB}$, this excess is $f_* - \mathrm{LB}$, which is larger than the objective error in 15 of the 16 Belgian runs. For the non-bundle methods SUBG, SUBG-EP, SUBG-L, FGM, DA, and DoWG, no such gap is available, so stopping is time-based.

### 4.5.5 Computational diagnostics

We now look closer at the per-iteration cost of each method and its sensitivity to the chosen parameters.

**Oracle and master step statistics.** We measure the time to solve each single-generator MILP subproblem. Here mean oracle time is the average time to solve one such subproblem. A full oracle evaluation solves one subproblem per generator. Results are summarized in Table 4.6.

Table 4.6: Single-generator MILP subproblem statistics by instance set: mean number of variables, mean number of constraints, mean solve time, and standard deviation of solve time.

| Set | # vars | # constrs | mean oracle (s) | std oracle (s) |
|---|---|---|---|---|
| BE | 488 | 748 | 0.0071 | 0.0074 |
| CA | 248 | 393 | 0.0021 | 0.0006 |

### 4.5.6 Sensitivity analyses

We next test the robustness of the conclusions with respect to numerical implementation choices and hyperparameter misspecification.

**Sensitivity to the smoothing parameter.** We ran FGM for 5 minutes on all 24 instances with $\varsigma \in \{10^{-9}, 10^{-8}, 10^{-7}\}$ and the step size $c\eta^*\varsigma/10^{-8}$, where $\eta^*$ is the step size tuned for $\varsigma = 10^{-8}$ and $c \in \{1/3, 1, 3\}$. For each instance and each $c$, let $\hat{\pi}_\varsigma$ be the point returned by FGM with smoothing parameter $\varsigma$, and let the relative spread be

$$\delta_c = \Big(\max_\varsigma f(\hat{\pi}_\varsigma) - \min_\varsigma f(\hat{\pi}_\varsigma)\Big) / \, |f^*|, \tag{4.41}$$

where the maximum and the minimum are taken over the three values of $\varsigma$. Table 4.7 reports the largest value of $\delta_c$ over each instance set.

Table 4.7: Largest value of $\delta_c$ (%) over each instance set, for each step-size factor $c$.

| | $c$ | | |
|---|---|---|---|
| Set | 1/3 | 1 | 3 |
| BE | $1.5 \times 10^{-3}$ | $1.6 \times 10^{-3}$ | $5.4 \times 10^{-3}$ |
| CA | $2.6 \times 10^{-4}$ | $8.8 \times 10^{-4}$ | $2.6 \times 10^{-3}$ |

A larger smoothing parameter degrades the convergence of FGM: with $\varsigma = 10^{-7}$, the mean objective error after 5 minutes is 4 to 12 times larger than with $\varsigma = 10^{-8}$, for every $c$ and on both instance sets. A smaller smoothing parameter, on the other hand, is not necessarily preferable, as it worsens the conditioning of the smoothed problem, whose gradient is $(1 + |\mathcal{G}|)/\varsigma$-Lipschitz. Indeed, although $\varsigma = 10^{-9}$ improves convergence on the Californian instances, it degrades it on the Belgian instances for some values of $c$. With $\varsigma = 10^{-8}$ and $c = 1$, FGM reaches the target accuracy on every instance within 5 minutes.

**Parameter sensitivity.** To assess sensitivity to parameter misspecification, we vary each method's tuned parameter over the grid used for tuning and measure the ratio between the worst and best mean objective errors across the grid. Objective errors below $10^{-6}$ times the reference value, the precision of the solver, are counted as $10^{-6}$ times the reference value. A ratio close to 1 indicates stability across parameter values, while a large ratio indicates sensitivity. We report one representative method per family: SUBG-EP for subgradient-type methods, PC-BPLM for bundle methods, DA for parameter-free methods, and FGM for smoothing-based methods.

Table 4.8: Worst-to-best ratio of mean objective error over the tuning grid, for one representative method per family.

| Family | Method | BE | CA |
|---|---|---|---|
| Subgradient-type | SUBG-EP | 19.41 | 16.38 |
| Bundle | PC-BPLM | 1.53 | 1.03 |
| Adaptive | DA | 2.88 | 12.04 |
| Smoothing | FGM | 64.29 | 20.54 |

Table 4.8 indicates that bundle methods are the least sensitive to parameter misspecification, followed by DA, whereas SUBG-EP and especially FGM exhibit higher sensitivity, most notably on the Belgian dataset.

## 4.6 Conclusion

This chapter studied a broad family of first-order methods applied to the dual formulation of CH pricing. All proposed methods are novel in the context of CH pricing except the subgradient method (SUBG) and the bundle level method (BLM). We also considered a smooth approximation of the dual CH pricing problem solved by FGM, an accelerated first-order method. To conduct a computational study of all presented algorithms, we provide the `ConvexHullPricing.jl` package and use it on 24 large-scale real-world instances. Subgradient methods, although apparently neglected by the community, are particularly competitive when equipped with good initialization and stepsize schedules, as illustrated by SUBG-EP and SUBG-L. The central finding is that exploiting the separability of the Lagrangian objective through clustered multi-cut under-approximations, combined with a variable-metric trust region, is the decisive algorithmic factor. Among all tested methods, the variable-metric multi-cut bundle methods (PC-BLM and PC-BPLM) clearly dominate on the Belgian instances and are among the fastest methods on the Californian instances. A systematic ablation (Table 4.3) shows that generator-wise multi-cut models alone yield little speedup, whereas PC-BLM and

PC-BPLM are 2.4 to 2.9 times faster than BLM. Column generation, the one primal method we retained as a reference point, reaches the target tolerance on every instance, but starts from a weaker position than the dual methods. Good initial prices are inexpensive to compute in the dual space, for instance from the continuous relaxation, whereas good initial columns would require feasible schedules close to the optimal ones, which essentially amounts to solving the UC problem. Our bundle implementations remain relatively simple; more refined variants, for instance with more elaborate management of the stabilization parameters or of the bundle (see Frangioni (2020)), could further improve the bundle family, so our conclusions in its favor are, if anything, conservative. Further analysis of decomposition methods applied to extended formulations (Paquet, 2021; Sadykov & Vanderbeck, 2013; Yu et al., 2020) is a natural direction for future work. In our implementation, we solve the generator subproblems in parallel and wait at each iteration until all subproblems are solved. Asynchronous level bundle methods (Iutzeler et al., 2020) update the cutting-plane model with each subproblem as soon as it returns, while keeping convergence guarantees; adapting them to CH pricing would remove this wait and potentially improve the practical performance of bundle methods.

Part II

# Systematic Frameworks for Optimization

# 5 GUARANTEE-BASED RECOMMENDATION OF FIRST-ORDER METHODS


## ABSTRACT

Choosing a first-order method for a given problem amounts to identifying a formulation to which a known convergence theorem applies. We present a disciplined modeling language that represents the algorithmic structure of a model, searches the equivalent formulations it supports, verifies the assumptions of known method templates, and returns the applicable methods together with their verified assumptions and convergence guarantees.


## BASED ON

*This chapter is based on joint work with Professor François Glineur. A user guide for the* `Argo.jl`[10] *package is under preparation.*

## IN THIS CHAPTER

First, Section 5.2 shows how different decompositions and a parametric reformulation of elastic-net logistic regression expose different applicable methods and worst-case guarantees. Section 5.3 then introduces structured objectives, the calculus of assumptions and oracles, and the registered reformulation rules used before theorem matching. Building on these descriptions, Section 5.4 explains how candidate formulations are matched to templates and their associated convergence theorems, while Section 5.5 compares worst-case guarantees under a fixed comparison context. Section 5.6 then reports computational examples: switching-time selection for hybrid methods, recursive matching of proximal subproblems, and the comparison of methods when oracle costs are unknown. Finally, Section 5.7 offers concluding remarks.

[10] https://github.com/SofianeTanji/PhD-Software/Argo.jl

## 5.1 INTRODUCTION

Algebraic modeling languages (Bussieck & Meeraus, 2004; Dunning et al., 2017; Fourer et al., 1990; Löfberg, 2004; Lubin et al., 2023) let the user state an optimization problem in mathematical notation and transform it into a form accepted by a solver. This principle is standard across mathematical programming, including nonlinear and mixed-integer settings (Jalving et al., 2022). In disciplined convex programming, this translation is governed by a grammar of atoms and composition rules: the grammar verifies that an expression is convex and determines how to reduce it to an equivalent solver-ready formulation (Agrawal et al., 2018; Diamond & Boyd, 2016; Grant & Boyd, 2014; Udell et al., 2014). Disciplined nonlinear programming uses a related grammar for a broader family of nonlinear models (Cederberg et al., 2026). In both cases, the grammar decides whether a model reduces to a prescribed standard form. We use a structured problem description for a different purpose: to work out which convergence theorems for first-order methods apply.

Since the advent of modern computational mathematics, the literature on optimization algorithms has been ever-growing, and a wide range of methods is now available. Published algorithms are usually shown to apply to given templates of optimization problems, and come with proofs characterizing their convergence rates. There exist many variations of such templates. A template may describe the objective as a single function or as a sum of functions with different characteristics (say, smoothness, convexity, existence of a computable proximal operator), accompanied by constraints that can themselves be described in various ways (linear constraints, a feasible set with a computable projection operator, functional constraints).

For a given instance, it is not always easy to identify which templates it can match, especially if one wants to consider equivalent reformulations of the problem. An unconstrained problem with an indicator term can be rewritten with an explicit constraint, a smooth term can be attached to either half of a composite objective, and curvature can sometimes be redistributed across terms without changing the function being minimized. All equivalent as mathematical objects; not at all equivalent as inputs to an algorithm. Choosing a method thus comes down to finding a formulation that some existing theorem already covers, which makes the task tedious for the practitioner. Given a specific formulation of an optimization problem, identifying formulations amenable to known convergence results typically requires applying a sequence of transformations until one of the recognized templates is reached. This reformulation process is both time-consuming and error-prone, and its effectiveness depends on the transformations and templates known to the user.

Moreover, not all reformulations are equally useful. Guarantees for a reformulated problem may or may not transfer back to the original problem, depending on the type of reformulation and the performance measure being considered. Even when two guarantees both transfer back, they are not necessarily comparable: for example, one may bound the objective gap and another the gradient norm. Identifying applicable methods therefore also requires fixing a *comparison context*: the performance measure (e.g., objective gap or gradient norm), the iterate convention (e.g., best, last, or average iterate), the initial bounds and their values (e.g., on the initial distance to the solution set or the initial objective gap), problem constants, and budget, together with oracle costs when relevant.

Our goal is to provide a disciplined modeling language for first-order methods. Such a language should represent the algorithmic structure of a model, search the formulations it supports, verify the assumptions of known method templates, and return applicable methods together with their verified assumptions and convergence guarantees.

### 5.1.1 Our approach

In this chapter, we present an automated approach that maps a structured optimization model to first-order algorithmic templates and, where their worst-case guarantees are comparable, ranks the resulting certificates accordingly. Specifically, we address the following question:

*Given a structured optimization problem, which templates does it satisfy, possibly after reformulation, which methods are applicable, and how do their comparable worst-case guarantees rank?*

A complete answer includes the optimization method, the convergence result being used, the formulation of the problem to which this result is applied, and the assumptions that make this application valid. When the method is obtained through a reformulation, the answer should also specify how the resulting guarantee transfers back to the original problem.

Worst-case analysis provides a reliability criterion, a theoretical basis for comparison, and a principled means of filtering methods prior to instance-level benchmarking. It is essential in critical settings where guarantees are required. The recent development of tight worst-case analyses is particularly relevant for the comparative ranking objective described above. In particular, the performance estimation problem (PEP) framework introduced in Drori & Teboulle (2014) and Taylor et al. (2017a) has enabled the derivation of exact or tight worst-case convergence rates for many first-order methods and function classes. The convergence results used in our framework may come from classical analyses or from such

PEP-based results, provided that their assumptions and performance measures are explicitly stated.

Our approach, illustrated in Figure 5.1, proceeds as follows:

1. represent an oracle-based optimization problem in terms of its objective terms, structural properties, constants, and available oracles;
2. automatically generate a structured set of reformulations of the given problem;
3. check whether each formulation satisfies one or several templates from a predefined catalogue of convergence results;
4. identify first-order methods with proven convergence guarantees for these templates;
5. construct a certificate for each recommendation, recording the formulation, the theorem used, the required properties and oracles, and the status of the reformulation path;
6. rank certified recommendations by their worst-case bound;

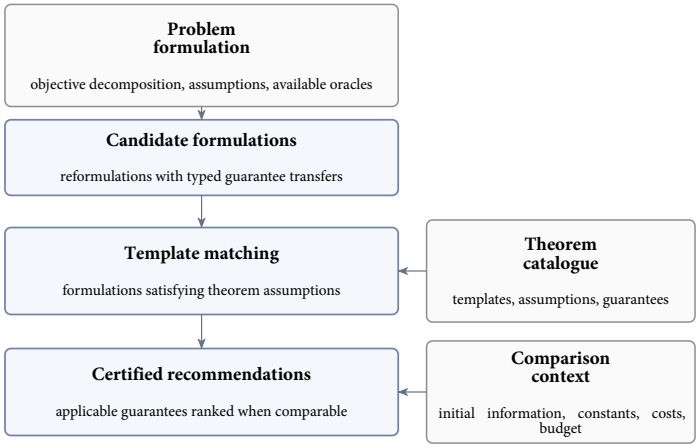



Figure 5.1: From a structured problem formulation to certified first-order methods in `Argo.jl`

### 5.1.2 Our contributions

Our main contributions are as follows:

1. We introduce a disciplined modeling framework for first-order methods. It represents oracle-based optimization problems together with their structural assumptions, admissible reformulations, method templates, and convergence certificates.

2. We extract from the literature a curated set of convergence theorems for classical first-order methods. For each theorem, we explicitly identify the problem class, required oracles, quantity being bounded, output iterate convention (last iterate, average iterate …), assumptions on the initial point, convergence rate, and bibliographic source. Each convergence result is thereby transformed into a reusable template against which a problem formulation can be verified.
3. We design a procedure that matches a given optimization problem, including formulations reached through reformulation steps, against templates in this set. Each reformulation step is typed according to the guarantee it can transfer to the preceding formulation. By tracking these steps, we can establish whether and how a certificate obtained for a reformulated problem applies to the original problem.
4. We provide a procedure that returns certified first-order methods from the resulting matches and ranks them by worst-case performance whenever their guarantees are comparable.
5. We show how the framework supports three extensions beyond single-method recommendation:
   1. **Parametric reformulations.** When an exact reformulation admits a free parameter, that parameter may be chosen to minimize the worst-case bound over the resulting family of equivalent formulations.
   2. **Hybrid methods.** For methods that switch once between two algorithms, the framework selects the switching time by minimizing the bound obtained from combining two compatible convergence guarantees.
   3. **Controlled-inexact proximal oracles.** For methods whose guarantees require a controlled-inexact proximal oracle, the framework recursively certifies a method for each proximal subproblem at the required accuracy.
6. We release `Argo.jl`, an open-source Julia package implementing the complete pipeline described above.

### 5.1.3 Related work

**Algebraic modeling languages and automatic transformations.** Algebraic modeling languages such as CVX (Grant & Boyd, 2014), YALMIP (Löfberg, 2004), GAMS (Bussieck & Meeraus, 2004), AMPL (Fourer et al., 1990), JuMP (Lubin et al., 2023), and related systems represent optimization problems symbolically and transform them into a form accepted by a solver. This transformation procedure is closely related to ours at the modeling level: both approaches manipulate structured mathematical expressions and rely on rules that preserve the meaning of the problem. The objective, however, is different. Modeling languages usually canonicalize a problem into a solver-ready form, such as a conic, quadratic, nonlinear,

or mixed-integer representation. We instead keep track of several formulations of the problem, match each to the first-order method templates it satisfies, and compare the resulting worst-case guarantees under a common comparison context. Conversely, the expression graph built by such a language, for instance a JuMP model, could serve as an input to `Argo.jl`, provided that its subexpressions are matched to atoms with known properties and oracles.

Many frameworks apply successive transformations to an initial problem so that a more efficient method becomes applicable. In general modeling languages, the goal is to match the problem to the standard form required by a solver chosen by the user. Similar frameworks exist in continuous optimization with first-order methods. In convex signal processing tasks, TFOCS (Becker et al., 2011) performs transformations before applying an optimal first-order method to the modified problem. This approach is similar to ours, in essence, as its goal is to use the method with the best theoretical guarantees on an optimized formulation, but it is tailored to specific signal processing problems. Moreover, its transformation process is systematic and always yields a *single* standard form, to which a *single* method is applied. In contrast, our framework explores *many* reformulations and ranks *many* methods by worst-case performance.

**Benchmarking in optimization.** The method selection problem is often addressed empirically through benchmarking. Benchmarking consists of solving a set of representative problem instances with several optimization methods and comparing their practical performance; when done carefully, it provides valuable information about robustness, implementation quality, and finite-time behavior. See Bartz-Beielstein et al. (2020) for best practices and Moreau et al. (2022) for a reproducible Python framework. Our approach is complementary and operates before any instance is solved. We start from the structural assumptions of the problem and a theorem catalogue, recommend the methods whose assumptions match the formulation, and consider reformulations of the problem that may bring further methods into scope or yield sharper guarantees.

**Aggregation and structuring of optimization knowledge.** Several recent works aim to organize optimization algorithms and their properties in a more systematic way. For example, Linnaeus (Lessard & Udell, 2025; Zhao et al., 2021) is a computer-algebra system that checks whether two iterative optimization methods are equivalent by analyzing their transfer functions. It also serves as a searchable repository of optimization methods, allowing users to identify whether a method is equivalent to one already present in the Linnaeus library. However, it focuses on equivalence testing of methods whereas `Argo.jl` certifies the applicability of

methods to an optimization problem. In other words, a Linnaeus user starts from an algorithm and asks whether it is equivalent to a known one, whereas a `Argo.jl` user starts from an optimization problem and asks which methods are certified for it and how their guarantees compare.

**Automated worst-case complexity analysis.** The framework we propose relies heavily on worst-case complexity bounds to make recommendations. The performance estimation problem (PEP) framework provides a powerful way to compute exact or tight worst-case bounds for fixed algorithms, function classes, and performance measures by solving semidefinite programs (Drori & Teboulle, 2014; Taylor et al., 2017a). Software such as the MATLAB and Python implementations of PEP (Goujaud et al., 2024; Taylor et al., 2017b) makes these computations accessible. The role of PEP is therefore upstream of ours: it can provide sharp worst-case bounds for particular algorithms, problem classes, and performance measures. In `Argo.jl`, such bounds are taken as convergence results from the literature, together with their assumptions, and are used to decide which methods apply to a given formulation and how their certified rates compare.

**Notation.** All vector spaces are finite-dimensional Euclidean spaces. Unless stated otherwise, norms and dual norms are Euclidean; smoothness, Lipschitz, strong-convexity and diameter constants are interpreted with respect to this convention. We write $\overline{\mathbb{R}} = \mathbb{R} \cup \{+\infty\}$. For a set $C$, its indicator is

$$\delta_C(x) = \begin{cases} 0 & x \in C, \\ +\infty & x \notin C. \end{cases} \tag{5.1}$$

## 5.2 A guided tour of `Argo.jl`

### 5.2.1 Formulations of the elastic-net problem

We use elastic-net logistic regression to follow, step by step, the reasoning that `Argo.jl` automates. This is a simple problem which admits several exact decompositions, each exposing different assumptions, oracles, and convergence guarantees. We show the corresponding Julia code for the guided tour at the end of the section, and we provide a complete example in the `Argo.jl` repository.

Consider data pairs $\{(a_i, b_i)\}_{i=1}^n$, with feature vectors $a_i \in \mathbb{R}^d$ and labels $b_i \in \{\pm 1\}$. The optimization variable is $x \in \mathbb{R}^d$. Let $\ell$ denote the empirical logistic loss.

$$\ell(x) = \frac{1}{n} \sum_{i=1}^{n} \log\Big(1 + \exp\Big(-b_i a_i^\top x\Big)\Big). \tag{5.2}$$

For regularization parameters $\lambda_1 > 0$ and $\lambda_2 > 0$, let $F$ denote the elastic-net logistic objective.

$$\min_{x \in \mathbb{R}^d} F(x), \quad F(x) \stackrel{\text{def}}{=} \ell(x) + \lambda_1 \|x\|_1 + \frac{\lambda_2}{2}\|x\|_2^2. \tag{5.3}$$

The first term is the empirical logistic loss, the second promotes sparsity, and the third adds strong convexity. We assume that a smoothness bound, denoted by $L_\ell$, is known for $\ell$ and that value and gradient oracles for $\ell$ are available.

Let $A$ be the data matrix with rows $a_i^\top$. The logistic loss is convex and smooth; in the Euclidean norm one may take $L_\ell = \|A\|_2^2/(4n)$. The function $x \mapsto \lambda_1\|x\|_1$ is globally Lipschitz with constant $\lambda_1\sqrt{d}$ and has an exact proximal operator. The function $x \mapsto \lambda_2\|x\|_2^2/2$ is smooth and strongly convex with constant $\lambda_2$, and it also has an exact proximal operator. The quadratic term makes $F$ fail to be globally Lipschitz on $\mathbb{R}^d$.

We use four exact decompositions of the same objective. Putting all terms in one bracket gives the one term formulation.

$$F(x) = \left[\ell(x) + \lambda_1\|x\|_1 + \frac{\lambda_2}{2}\|x\|_2^2\right]. \tag{5.4}$$

Grouping the loss with the quadratic regularizer gives the first two term formulation.

$$F(x) = \left[\ell(x) + \frac{\lambda_2}{2}\|x\|_2^2\right] + \left[\lambda_1\|x\|_1\right]. \tag{5.5}$$

Grouping the two regularizers gives the second two term formulation.

$$F(x) = [\ell(x)] + \left[\lambda_1\|x\|_1 + \frac{\lambda_2}{2}\|x\|_2^2\right]. \tag{5.6}$$

Keeping the original summands separate gives the three term formulation.

$$F(x) = [\ell(x)] + \left[\lambda_1\|x\|_1\right] + \left[\frac{\lambda_2}{2}\|x\|_2^2\right]. \tag{5.7}$$

Brackets show which terms are treated together in the theorem. All four formulations define the same function and differ in the properties (smoothness, Lipschitz continuity, strong convexity) and oracles (proximal operators, gradients, subgradients) they expose.

In the one term formulation (5.4), $F$ is treated as one nonsmooth strongly convex function. Classic literature on subgradient methods requires global Lipschitz continuity, which is not satisfied by $F$. However, $F$ is locally Lipschitz around its minimizer, and Grimmer (2019) gives a deterministic $\mathcal{O}\left(1/\sqrt{k}\right)$ best iterate bound on the suboptimality gap.

For the two term formulation (5.5), the first bracket is $(L_\ell + \lambda_2)$-smooth and $\lambda_2$-strongly convex, while the second bracket is convex and proximable. Hence the formulation satisfies the composite theorems with one smooth strongly convex term and one convex proximal term.

In the alternative two term formulation (5.6), it is the proximal term which carries the strong convexity. Denote the proximal term by $g$, so

$$g(x) = \lambda_1 \|x\|_1 + \frac{\lambda_2}{2}\|x\|_2^2. \tag{5.8}$$

The function $g$ is $\lambda_2$-strongly convex and has a closed-form proximal operator that is inexpensive to compute. This decomposition is suited to composite theorems that allow strong convexity in the proximal term or in the full sum. If strong convexity must belong to the smooth term, the applicable formulation is (5.5).

The three term formulation (5.7) keeps the regularizers separate. It matches three-operator splitting templates such as Davis–Yin splitting (Davis & Yin, 2017), with

$$f = \ell, \quad g(x) = \lambda_1 \|x\|_1, \quad h(x) = \frac{\lambda_2}{2}\|x\|_2^2. \tag{5.9}$$

Here $f$ is smooth, while $g$ and $h$ have proximal operators.

Note that we only considered four decompositions. Indeed, an exact decomposition is useful only when its terms provide the assumptions and oracles required by a theorem. For example, the following decomposition is exact, but it does not match any theorem in our theorem catalogue.

$$F(x) = \left[\ell(x) + \lambda_1 \|x\|_1\right] + \left[\frac{\lambda_2}{2}\|x\|_2^2\right]. \tag{5.10}$$

The first bracket in (5.10) would need an exact proximal operator, which is not available.

So far, we have considered only different groupings of the same terms. More generally, one may consider parametric reformulations, where a free parameter is introduced to generate a family of equivalent formulations. A simple example is

curvature transfer, which redistributes the curvature between different terms in the objective. Let the free parameter be $\rho$, with $0 \leqslant \rho \leqslant \lambda_2$. Applied to the alternative two term formulation (5.6), this gives the following reformulation.

$$F(x) = \left[\ell(x) + \frac{\rho}{2}\|x\|_2^2\right] + \left[\lambda_1\|x\|_1 + \frac{\lambda_2 - \rho}{2}\|x\|_2^2\right]. \tag{5.11}$$

The endpoint $\rho = 0$ gives (5.6), while the endpoint $\rho = \lambda_2$ gives (5.5). Interior values of this interval split the quadratic curvature between the smooth and proximal terms. For a fixed theorem, only the values of $\rho$ satisfying its assumptions are kept. Varying $\rho$ changes the constants entering the bound, and the best value may depend on the theorem.

The same split applied to the three term formulation gives the following exact reformulation.

$$F(x) = \left[\ell(x) + \frac{\rho}{2}\|x\|_2^2\right] + \left[\lambda_1\|x\|_1\right] + \left[\frac{\lambda_2 - \rho}{2}\|x\|_2^2\right], \quad \rho \in (0, \lambda_2). \tag{5.12}$$

Again the objective is unchanged, while the convergence rate of an applicable method may vary through the smoothness and strong convexity constants.

### 5.2.2 Comparable guarantees

Let us now fix the comparison context. Let $k$ denote the iteration budget, let $F^*$ denote the minimum value of $F$, and let $x_k$ denote the last iterate after $k$ iterations. Suppose that the requested guarantee is an upper bound on the last-iterate gap $F(x_k) - F^*$, and that the initial information is an upper bound $R^2$ on the squared distance from the initial point to a solution. With this guarantee and comparison context fixed, the one term formulation (5.4) yields a non comparable certificate. Indeed, the only applicable theorem for the one term formulation is due to Grimmer (2019), which gives a guarantee for the best iterate but none for the last iterate. Similarly, the three term formulation (5.7) only yields methods with ergodic guarantees. The relevant candidates are therefore the two term formulations (5.5), (5.6), and the exact family (5.11). For each applicable theorem, the parameter $\rho$ is chosen by `Argo.jl` to minimize the corresponding worst-case bound over the admissible part of this family. Thus, we have reduced the original problem to a list of candidates for which we can have the requested guarantee. `Argo.jl` does this automatically, and uses its theorem catalogue to recommend the best methods for the given comparison context.

### 5.2.3 The example in Julia

In the previous section, we explained how to take a structured optimization problem, consider its decompositions and reformulations, and identify applicable convergence theorems. In the following lines, we show how to perform the same steps in Julia using the `Argo.jl` package.

**Imports.** We import the package as follows.

```
using Argo
using Argo.CurvatureTransfer
```

Here, `Argo` is the main package, and `Argo.CurvatureTransfer` provides the curvature transfer reformulation rule, which is not part of the core module.

**Specifying the problem.** We can now specify the elastic-net logistic regression problem in `Argo.jl`. We need to declare a variable, the objective terms, and the problem to be minimized.

```
x = variable(:x)

L_ℓ = symbolic(:L_ℓ)
M₁ = symbolic(:M₁)
λ₂ = symbolic(:λ₂)

ℓ = term(
    :ℓ,
    x;
    properties=[convex(), smooth(L_ℓ)],
    oracles=[oracle(:value), oracle(:gradient)],
)
r₁ = term(
    :r₁,
    x;
    properties=[convex(), lipschitz(M₁)],
    oracles=[oracle(:value), oracle(:subgradient),
oracle(:prox)],
)
r₂ = term(
    :r₂,
    x;
    properties=[
        convex(),
        smooth(λ₂),
        strongly_convex(λ₂),
        centered_quadratic(λ₂),
    ],
    oracles=[oracle(:value), oracle(:gradient), oracle(:prox)],
)

problem = minimize(ℓ + r₁ + r₂)
```

Note that the logistic loss as well as the $\ell_1$ and $\ell_2^2$ regularizers are atoms in `Argo.jl`, meaning they are already predefined with their properties and oracles in `Argo.jl` and we do not need to define them from scratch. They can be used by importing the existing atoms in `Argo.jl`, which is done by `Argo.Atoms`. Here, for illustration purposes, we show how to define them from scratch.

**Specifying the comparison context.** We now provide the numerical initial information, problem constants, and iteration budget under which the bounds will be compared. This supplies the numerical part of the comparison context.

```
k = 100
R² = 1.0

context = ComparisonContext(
    initial_bounds=Dict(:distance_squared => R²),
    values=Dict(:L_ℓ => 9.0, :M₁ => 1.0, :λ₂ => 1.0),
    max_iterations=k,
)
request = FixedBudgetRequest(k, context)
```

**Computing applicable guarantees.** We can then ask `Argo.jl` to produce certificates for the requested guarantee quantity, keep those with the required output and form of initial information, and rank them under the comparison context.

```
certified = certificates(
    problem;
    quantity=:objective_gap,
    reformulations=[CurvatureTransferRule()],
    reformulation_depth=1,
)

target_certificates = filter(certified) do certificate
    guarantee = certificate.guarantee
    return guarantee.output === :last &&
           guarantee.initial_bound === :distance_squared
end

report = rank(target_certificates, request)
```

Here we compare bounds on the last-iterate objective gap after 100 iterations, assuming $\|x_0 - x^*\|_2^2 \leqslant 1$ and fixing $L_\ell = 9$, $M_1 = 1$, and $\lambda_2 = 1$. The `quantity` argument and the certificate filter select the performance measure, iterate convention, and form of initial bound, while `ComparisonContext` supplies the numerical values used to rank the bounds.

We list the resulting comparable certificates in Table 5.1

Table 5.1: Comparable entries returned by `Argo.jl` for the elastic-net example.

| method | formulations tested |
|---|---|
| FISTA (Beck & Teboulle, 2009) | (5.6), (5.5), and admissible values of (5.11) |
| SC-FISTA[11] (Lin & Xiao, 2014) | (5.5) and admissible values of (5.11) |
| SAPG (Taylor et al., 2017c) | (5.6), (5.5), and admissible values of (5.11) |

**Ranking certified bounds.** We can also rank these certificates numerically by their worst-case bound, once the constants $L_\ell$, $\lambda_1$, $\lambda_2$, and $R^2$ are fixed. In this example, up to a common factor, the certified bounds depend on the constants only through the condition number $\kappa = (L_\ell + \lambda_2)/\lambda_2$ and the iteration budget $k$. We fix $\lambda_2 = 1$ and $R^2 = 1$, and vary both $L_\ell = \kappa - 1$ and the number of iterations $k$, each over 161 logarithmically spaced points, respectively from 1 to $10^8$ and 1 to $10^4$. At each cell of this grid, `Argo.jl` ranks the certified bounds for FISTA, SAPG, and SC-FISTA, and records the method attaining the smallest one. SC-FISTA is unavailable at $\kappa = 1$, while all three certificates are compared in the remaining cells. The resulting regime diagram is shown in Figure 5.2.

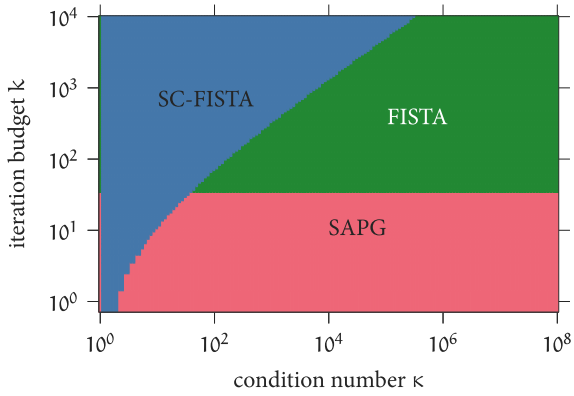



Figure 5.2: Regions in the $(\kappa, k)$ plane where the smallest certified upper bound on $F(x_k) - F^*$ is attained by the indicated method, with the remaining constants fixed.

[11] SC-FISTA and FISTA differ in their choice of momentum sequence.

**Interpreting the regimes.** Although SC-FISTA is designed for strongly convex problems, it is not always the best choice: FISTA and SAPG provide better guarantees for sufficiently ill-conditioned problems, SAPG for small iteration budgets and FISTA for larger ones. Indeed, the linear rate of SC-FISTA contracts by the factor $1 - 1/\sqrt{\kappa}$ per iteration, so for large $\kappa$ its bound stays near its initial value during roughly the first $\sqrt{\kappa}$ iterations, while the bounds of FISTA and SAPG already decrease as $1/k^2$.

## 5.3 Problem model and reformulations

### 5.3.1 Structured objectives

We represent an optimization problem as a structured objective built from named functions, which we call terms, and which can be composed through simple operations, listed in Table 5.2, to form the overall objective.

Table 5.2: Operations for building objective terms in `Argo.jl`

| Operation | Form |
|---|---|
| Sum | $f + g$ |
| Difference | $f - g$ |
| Scaling | $\alpha f$ |
| Composition | $f \circ g$ |
| Pointwise maximum | $\max(f, g)$ |

For example, a problem of the form

$$\min_x f(x) + g(Ax) + h(x) \tag{5.13}$$

is declared by naming the variable $x$, the linear map $A$, and the functions $f$, $g$, and $h$, and then forming the displayed objective.

```
using Argo

problem = @model begin
    @variable x[1:n]

    @term A
    @assume A linear

    @term f
    @term g
    @term h

    @minimize f(x) + g(A(x)) + h(x)
end
```

This keeps the model close to the mathematical expression while making explicit which functions are available to the convergence theorems used later. At this stage, only the expression[12] of the problem is declared. The regularity assumptions, constants, and available oracles attached to these functions are separate pieces of information, discussed in detail in the next subsection. We also point out that this macro-based notation differs from the functional syntax used in the guided tour (Section 5.2.3): both are valid ways of specifying a problem in `Argo.jl`.

### 5.3.2 Calculus of assumptions and oracles

In this subsection, we present two key abstractions in `Argo.jl`: assumptions and oracles. An assumption is a property of a function that is required by a convergence theorem, such as convexity, smoothness, or strong convexity. Functions are supposed to only be accessible through an oracle, which can be queried at points of the domain to obtain information about the function, such as its value or gradient.

Once functions are introduced in `Argo.jl`, it is crucial that the user states the assumptions each term carries as well as the oracles he can access, as these are the only information that `Argo.jl` uses to match the problem to convergence theorems of methods.

The main oracles present in `Argo.jl` are listed in Table 5.3. Projection is represented as a proximal oracle for an indicator function. If the user doesn't have access to an exact oracle but only to an inexact oracle (low-precision, approximate solution of a subproblem, etc.), he can still declare it in `Argo.jl`, and the framework will only match the problem to convergence theorems that allow inexact oracles.

[12] In reality, we also declared the information that $A$ is a linear map. This could be omitted here and stated later on, along with the regularity assumptions of the other terms.

Table 5.3: Main oracle types used by `Argo.jl`

| Oracle | Mathematical operation | Julia symbol |
|---|---|---|
| Evaluation | compute $f(x)$ | `:value` |
| Gradient | compute $\nabla f(x)$ | `:gradient` |
| Subgradient | choose $s \in \partial f(x)$ | `:subgradient` |
| Proximal | solve $\arg\min_y f(y) + \frac{\|y-x\|^2}{2\gamma}$ | `:prox` |
| Linear minimization | solve $\arg\min_{y \in C} d^T y$ | `:linear_minimization` |

Assumptions are often accompanied by constants, which are numerical values that quantify the property: for example, a smoothness constant $L$ or a strong convexity constant $\mu$. In `Argo.jl`, these constants can be represented either as numerical values if known or symbolic parameters if they are unknown. In the guided example (Section 5.2), $L_\ell$, $\lambda_1\sqrt{d}$, and $\lambda_2$ remained symbolic until we fixed them to specific numerical values to produce the regime diagram (Figure 5.2). These numerical values are part of the comparison context, since worst-case bounds cannot be ordered before the constants entering them are fixed. Constants can either be supplied directly by the user or provided within the `Argo.jl` for atom functions (such as standard norms) or derived from other constants by the calculus presented below. The assumptions and oracles of a user-defined term are declared directly in the model block. For example, the following model states that $f$ is convex and smooth with constant $L_f$, while $g$ is convex and Lipschitz with constant $M_g$. It also states that value and gradient oracles are available for $f$, and that value, proximal, and subgradient oracles are available for $g$.

```
problem = @model begin
    @variable x[1:n]

    @term f
    @assume f convex smooth(:L_f)
    @oracle f value gradient

    @term g
    @assume g convex lipschitz(:M_g)
    @oracle g value prox subgradient

    @minimize f(x) + g(x)
end
```

Oracle declarations may also carry costs. For instance, the following declaration assigns a unit cost to each evaluation of $f$.

```
problem = @model begin
    @variable x[1:n]

    @term f
    @assume f convex smooth(2.0)
    @oracle f value(cost=1.0)

    @minimize f(x)
end
```

An important feature of `Argo.jl` is that it can automatically derive the properties and oracles of composite terms from those of their constituents.

We give below simple rules for the most common assumptions, which are implemented in `Argo.jl`. All properties and constants are understood with respect to the same norm. A nonnegative linear combination of convex functions is convex. If each $f_i$ is $\mu_i$-strongly convex, allowing $\mu_i = 0$ for a merely convex summand, then $\sum_i f_i$ is $\left(\sum_i \mu_i\right)$-strongly convex. Smoothness and Lipschitz constants add over sums, and multiplication by a positive scalar multiplies the corresponding constants. If $h$ is convex, then $x \mapsto h(Ax)$ is convex for every linear map $A$. If $h$ is $L_h$-smooth and $\|A\| \leqslant M$, then this composition is $L_h M^2$-smooth. If $h$ is $G_h$-Lipschitz, then the composition is $G_h M$-Lipschitz. Finally, if $h$ is $\mu_h$-strongly convex and $\|Ax\| \geqslant m\|x\|$ for every $x$, then $x \mapsto h(Ax)$ is $\mu_h m^2$-strongly convex.

Similarly, we provide common rules for combining oracles. A sum has an evaluation or gradient oracle when every summand has the corresponding oracle. For convex functions, if $s_i \in \partial f_i(x)$ is available for every $i$, then

$$\sum_i s_i \in \partial\left(\sum_i f_i\right)(x). \tag{5.14}$$

For a differentiable function composed with a linear map, the gradient follows from

$$\nabla(h \circ A)(x) = A^\top \nabla h(Ax). \tag{5.15}$$

Proximal oracles do not combine in general. Knowing the proximal operators of $f$ and $g$ does not give the proximal operator of $f + g$. One useful exception is the addition of a centered quadratic. To state this exception, we denote the proximal operator of $g$ with parameter $\gamma$ by $\operatorname{prox}_{\gamma g}$, as in Section 2.1. For $a \geqslant 0$, let $q_a(y) = a\frac{\|y\|_2^2}{2}$. Then, for every $\gamma > 0$,

$$\operatorname{prox}_{\gamma(g+q_a)}(x) = \operatorname{prox}_{\left(\frac{\gamma}{1+a\gamma}\right)g}\left(\frac{x}{1+a\gamma}\right). \tag{5.16}$$

Thus, adding a centered quadratic to a proximable function preserves proximal access after a change of centre and proximal parameter.

When an oracle is obtained from several constituent oracles, its accuracy cannot be stronger than that of the oracles used to evaluate it. Its cost also includes the corresponding constituent evaluations. No property or oracle is inferred unless it follows from one of these rules or is supplied directly.

### 5.3.3 Decomposition and reformulations

The previous subsections describe how a problem is written and how assumptions, constants, and oracles are derived for its terms. We now describe the reformulation pool used before theorem matching. Starting from the original problem, `Argo.jl` applies the registered reformulation rules and adds the resulting candidates to the pool. The original problem is part of this pool. In the implementation, the objective is represented as an expression tree whose nodes use a custom Julia struct called Term. Leafs are the named functions, while internal nodes are the operations used to combine them. Each node also carries oracles and properties, either declared by the user or derived from the calculus of assumptions and oracles. The search proceeds breadth-first, using a set to avoid revisiting the same formulation with the same requested guarantee quantity. These structures use standard Julia collections, without a graph library. The pool is finite because the set of rules and their application depth are fixed. Each candidate is a mathematical formulation: an objective expression with assumptions, constants, and available oracles for its terms, together with a relation to the original problem.

Grouping is the simplest exact reformulation rule. It names selected subexpressions and combines them using the calculus of Section 5.3.2. In Section 5.2, groupings were only used to combine summands, but the same idea applies to any operation in Table 5.2. For instance, if $F(x) = f(g(x)) + h(x)$, one may introduce the term $u = f \circ g$ and formulate the decomposition $F = u + h$. Thus the same objective may enter the pool through several exact formulations exposing different assumptions and oracles.

In the current software implementation, grouping is not represented as a reformulation rule[13]. Instead, the theorem matcher automatically examines the relevant groupings while assigning objective terms to the functions of a theorem template.

[13] This is mostly due to historical developments of the toolbox. Technically, it could also be implemented as a reformulation rule.

Curvature transfer is another reformulation technique. It moves curvature from one term to another by subtracting and adding the same centered quadratic. We illustrate this with a simple example.

Let $f$ denote a $\mu$-strongly convex term with $\mu > 0$, let $g$ denote a convex term, and let the objective be $F = f + g$. For any transfer parameter $\rho$ with $0 \leqslant \rho \leqslant \mu$, we use the following exact reformulation.

$$F(x) = \left[f(x) - \frac{\rho}{2}\|x\|_2^2\right] + \left[g(x) + \frac{\rho}{2}\|x\|_2^2\right]. \tag{5.17}$$

The first term is $(\mu - \rho)$-strongly convex, and the second term is $\rho$-strongly convex. The full amount of curvature is unchanged, while both terms may now satisfy strong-convexity requirements if imposed by any convergence theorem. This reformulation hence yields a parametrized family of reformulations. The numerical value of $\rho$ is chosen in a later step, after a theorem has matched and a comparable bound has to be evaluated, see Section 5.6.

Each reformulation step stores a typed guarantee transfer and an output map. To illustrate, consider the reformulation of a convex objective $F$ through its Moreau envelope $F_\lambda(x) = \min_y \left\{F(y) + \frac{1}{2\lambda}\|y - x\|_2^2\right\}$, and let $p_\lambda(x)$ denote a corresponding proximal point. The envelope is smooth, with $\nabla F_\lambda(x) = \frac{x - p_\lambda(x)}{\lambda}$. Thus, the proximal oracle for $F$ provides a gradient oracle for $F_\lambda$, which can make first-order methods for smooth problems applicable. The transfer specifies which guarantee must be established on the reformulated problem and how its bound yields the requested guarantee on the preceding formulation. Suppose that the requested guarantee on the preceding formulation is $F(y) - F^* \leqslant \varepsilon$. For the Moreau envelope, the reformulated problem must establish $F_\lambda(x) - F^* \leqslant \varepsilon$. In this case, the transfer maps an objective-gap guarantee for $F_\lambda$ to an objective-gap guarantee for $F$. The output map specifies which point to return to the preceding formulation. Here, the map sends $x$ to $y = p_\lambda(x)$. The definition of the envelope gives $F(p_\lambda(x)) - F^* \leqslant F_\lambda(x) - F^*$, so the envelope bound yields $F(y) - F^* \leqslant \varepsilon$.

Several reformulations may be applied successively. Their typed guarantee transfers are composed in reverse order, while the output maps record how the final point is carried back through the reformulation path. At the end of this process, the original problem is linked to each candidate by a path of reformulations.

Matching then proceeds one candidate and one theorem at a time. For each candidate and each theorem in the theorem catalogue, `Argo.jl` checks whether the terms of the candidate fit the functions appearing in the theorem statement, using only the assumptions, constants, oracles and guarantee transfers attached to that

candidate. A pair is kept when these establish every requirement of the theorem and when the sequence of reformulation steps carries the requested conclusion back to the original problem. The next section describes this step.

## 5.4 Template applicability and theorem certification

In this section, we describe how `Argo.jl` matches candidate formulations with entries in its theorem catalogue. To that effect, we first describe how convergence theorems are formalized in `Argo.jl`, and then we explain how the applicability of a theorem is checked for each candidate formulation.

**Formalizing convergence theorems.** In the previous section, we described how `Argo.jl` produces a pool of candidate formulations for a given problem. To find which optimization methods apply to each formulation in an automated manner, it is necessary to have a standard formalism for convergence theorems, so that their assumptions can be checked. We illustrate this point with a convergence theorem for the gradient method with fixed stepsize.

**Proposition 5.1.** (Fixed-step gradient descent on smooth convex functions) *Let* $f$ *be a convex differentiable function with* $L$*-Lipschitz gradient, and let* $f_*$ *denote its infimum. A gradient oracle can be queried at any point of* $\operatorname{dom} f$*. Let* $x_0$ *denote the initial point. Let* $N$ *denote a positive integer. Let* $\gamma > 0$ *be a stepsize such that* $\gamma L \in (0, 2)$*, and define the gradient iterates by*

$$x_{k+1} = x_k - \gamma \nabla f(x_k), \quad k = 0, \ldots, N-1. \tag{5.18}$$

*Then the following estimate holds (Rotaru et al., 2026):*

$$\|\nabla f(x_N)\|^2 \leqslant 2L \frac{f(x_0) - f_*}{1 + \gamma L \min\left\{2N, \frac{-1+(1-\gamma L)^{-2N}}{\gamma L}\right\}}. \tag{5.19}$$

In `Argo.jl`, a convergence theorem is specified by five elements: a template problem, a performance measure, initial information, an optimization method with conditions on its parameters, and a convergence rate.

The template problem is written with the modeling language of Section 5.3. In the example above, it is the minimization of a single convex function with a Lipschitz gradient, accessible through a gradient oracle, and is declared as follows.

```
using Argo
smooth_convex_template = @model begin
  @variable x
  @term f(x)
  @assume f convex smooth(:L)
  @oracle f gradient
  @minimize f(x)
end
```

For each assumption, it is important to name the constants that quantify the property, so that they can be referred to later in the theorem statement.

The performance measure is the quantity bounded by the theorem, while the initial information specifies the bound assumed at the starting point. Here the estimate controls the squared gradient norm at the last iterate in terms of the initial suboptimality gap $f(x_0) - f_*$. Optimization methods come with their parameters and the assumptions they must satisfy; for the fixed-step gradient method, the only parameter is the stepsize $\gamma$, admissible when $\gamma L \in (0, 2)$. This admissible range must be written as a function of the constants of the template. These five elements define an entry in the theorem catalogue.

We separate the numerical data of a theorem into constants and parameters. The template constants are the quantities that appear in its assumptions and in its initial information, for example $L$, $\mu$, or $R^2$. Their admissible set $\mathcal{S}_T$ records the scalar constraints under which the theorem is valid, such as $L > 0$, $\mu > 0$, $\mu \leqslant L$, or a user-declared comparison interval such as $L \in [1, 10^4]$. For a fixed constant vector $s \in \mathcal{S}_T$ and budget $k$, the theorem also specifies the admissible method parameters $\mathcal{U}_T(k, s)$. In the example above, $\mathcal{U}_T(k, s)$ is the set of stepsizes $\gamma$ such that $\gamma L \in (0, 2)$. The convergence rate $B_T(k, s, \theta)$ is the numerical estimate certified by the theorem for any admissible method parameter $\theta \in \mathcal{U}_T(k, s)$.

After template matching, we obtain a retained candidate–theorem pair $q$. The pair identifies the matched formulation and the theorem entry. It also stores the substitutions that replace theorem constants by constants of the candidate problem. The parameter vector optimized during recommendation can be augmented during the applicability phase by reformulation parameters, for example by the curvature-transfer parameter $\rho$. We write $B_q(k, s, u)$ for the convergence rate attached to this retained pair, and $\mathcal{U}_q(k, s)$ for its admissible parameter set.

**Verification of applicability.** We now formalize how a candidate formulation is matched with a convergence theorem. A theorem is specified by the five elements stated in the previous paragraph: a template problem, a performance measure,

initial information, an optimization method with conditions on its parameters, and a convergence rate. We examine what the matching requires of each element in turn.

First, the user specifies the guarantee quantity to be certified. `Argo.jl` retains only the theorems whose guarantee can be carried back to this quantity through the recorded reformulation path. The output convention (last iterate, best iterate, ...) and the required form of initial information are attached to the resulting certificates and used later to select comparable guarantees. Then, the optimization method and its parameters play no role in the matching: they enter only the computation of the rate, which we treat in Section 5.5. So the matching reduces to comparing the candidate formulation with the template problem.

We then check, for each candidate term and the template function it corresponds to, that the assumptions derived for the term by the calculus of Section 5.3.2 imply the assumptions required by the theorem, and that the term provides the oracles required by the theorem.

We illustrate this template–candidate matching on the $\ell_2^2$-regularized elastic-net problem (5.5) in Figure 5.3.

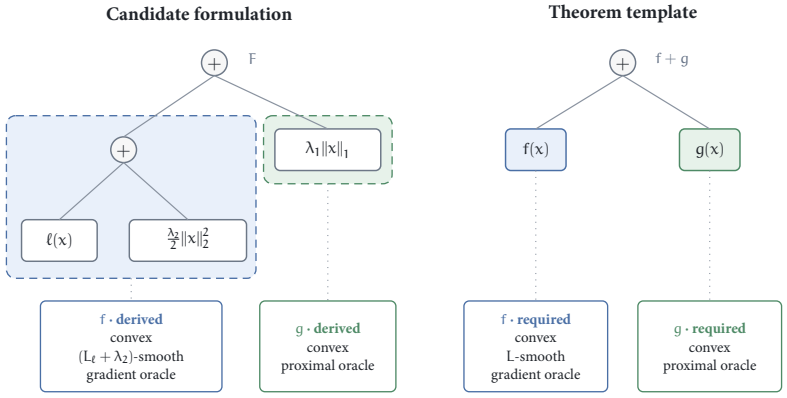



Figure 5.3: Matching of the elastic-net formulation (5.5) with the two-term proximal-gradient template. The colors show the correspondence: the smooth function f matches $\ell + \frac{\lambda_2}{2}\|\cdot\|_2^2$, and the proximal function g matches $\lambda_1\|\cdot\|_1$. The boxes compare the derived assumptions with the required assumptions. Here $L = L_\ell + \lambda_2$.

In the implementation, the correspondence shown by the colors is stored in a custom Julia `struct`, `RoleBinding`, as a vector of pairs associating theorem roles with objective expressions. The assumptions and oracles derived for those expressions are checked against the theorem requirements, as illustrated by the boxes. Each successful match produces a `Certificate` that identifies the theorem and

prescribed method, stores the role assignments, and specifies how the required oracles are supplied. It also stores the convergence guarantee and the reformulation steps used to carry that guarantee back to the original problem.

## 5.5 Method recommendation from comparable guarantees

In the previous section, we retained a pool of candidate–theorem pairs that establish the requested guarantee quantity. We now use their worst-case complexity bounds to recommend a method and a formulation. We recall that only certificates with the same performance measure and iterate convention can be compared. Their bounds are then evaluated under a common comparison context, which fixes the numerical value of the initial information, the problem constants, the cost assigned to each oracle call, and the budget. As these bounds are real-valued functions of the budget and the problem constants, they can only be numerically ordered once these quantities are fixed.

**Computing and ranking the bounds.** We first make precise what it means to evaluate a bound. Let $q$ denote a retained candidate–theorem pair. Let $k$ denote the budget, let $s$ collect the problem constants and the constants appearing in the initial information, and let $u$ collect the "free" method and reformulation parameters. Using the notation introduced in Section 5.4, the admissible parameters for $q$ form the set $\mathcal{U}_q(k, s)$ and the corresponding convergence rate is $B_q(k, s, u)$. The optimized bound is

$$B_q^*(k, s) = \min_{u \in \mathcal{U}_q(k,s)} B_q(k, s, u). \tag{5.20}$$

Every evaluation below includes this minimization. This means we compare methods in their best possible form, hence avoiding the discussions related to parameter tuning.

The vector $u$ contains both parameters of the method and parameters introduced by the formulation. For example, the fixed-step gradient theorem of Section 5.4 contains a free stepsize. For the curvature-transfer family (5.11), the free parameter is $\rho$. For each theorem, `Argo.jl` minimizes the bound over the part of $[0, \lambda_2]$ where its assumptions hold. The same convention is used at every numerical value of the constants, at every sample, and at every point of a regime diagram.

When the budget and all constants are numerical, we evaluate $B_q^*(k, s)$ for every retained pair $q$ and sort the values in increasing order. Note that the same method may appear several times depending on the formulation on which it is applied.

`Argo.jl` can also rank methods by *oracle work*: the sum of the declared costs of all oracle calls needed to reach the requested accuracy. When every call has unit cost, the oracle work is simply the number of calls to the oracles. It may be hard to provide specific values for each oracle. In that case, `Argo.jl` can report, for each method, the number of calls to each oracle used, and discard every method that needs at least as many calls to every oracle as another one, and strictly more calls to at least one of them. The remaining methods thus form a Pareto front, which may help decide which method to use, as illustrated in Section 5.6.3.

**Ranking with incomplete information.** When the constants are known only to belong to an admissible set $\mathcal{S}$, we summarize the comparison over numerical values drawn from this set. We fix the budget $k$, choose a probability distribution $\Pi$ supported on $\mathcal{S}$, and draw $m$ constant vectors $s_1, \ldots, s_m$. At each draw, we evaluate $B_q^*(k, s_j)$ and rank the same set of retained pairs.

Let $r_q^j$ denote the rank assigned to pair $q$ at draw $j$. Its mean rank is

$$|r|_q = \frac{1}{m} \sum_{j=1}^{m} r_q^j. \tag{5.21}$$

We order the pairs by increasing mean rank. The recommendation is heuristic and relative to $\Pi$: it depends on the sampling law, the number of draws, and the treatment of ties. The support of $\Pi$ must respect any relations among the constants.

The user supplies the distributions, the number of draws, and the random seed explicitly, so that the recommendation is reproducible.

The following example samples the unknown smoothness constant $L$ uniformly between 1 and 10 and ranks the applicable certificates after 100 iterations.

```
using Argo
using Argo.Sampling

f = term(
    :f;
    properties=[convex(), smooth()],
    oracles=[oracle(:gradient)],
)
applicable_certificates = certificates(minimize(f);
reformulation_depth=0)

sampled_ranking = Sampling.rank(
    applicable_certificates;
    distributions=Dict(:L => Sampling.Uniform(1, 10)),
    samples=2_000,
    seed=42,
    iterations=100,
    initial_bounds=Dict(:distance_squared => 1.0),
)
```

When the dependence on one or two quantities is itself of interest, they can be used as axes of a regime diagram. At each point of a given grid, `Argo.jl` evaluates the retained bounds and records every pair attaining the smallest value.

The guided tour gives such a diagram in Figure 5.2, with the condition number $\kappa$ and the iteration budget $k$ as axes. This representation retains the dependence of the recommendation on $(\kappa, k)$ without assigning a probability distribution to these quantities.

### 5.5.1 Empirical validation

Finally, we compare the certified recommendations with the behavior of the corresponding methods on two examples.

**Diagonal quadratics.** We focus on minimizing $F(x) = \frac{1}{2}(x - x^*)^T D(x - x^*)$ over $\mathbb{R}^{16}$, with $D = \mathrm{diag}(d_1, \ldots, d_{16})$. We fix the eigenvalues to be logarithmically spaced between $\mu = 1$ and $L = \kappa\mu$. In this simple setting, we can compute the suboptimality gap of applicable methods in closed form, and we can compare the certified recommendation against the method that actually reaches the smallest gap at iteration $k$. `Argo.jl` suggests five applicable methods: gradient descent (GD), Nesterov's accelerated gradient (AGD), simplified accelerated gradient (S-AGD), strongly convex accelerated gradient (SC-AGD), and a hybrid that runs S-AGD before switching to SC-AGD, with the switch time chosen by minimizing the overall convergence rate.

We evaluate `recommendation` over a $(\kappa, k)$ grid, 36 values of $\kappa \in [10, 10^6]$ by 32 values of $k \in [10, 10^5]$, and compare it against the method that actually reaches the smallest gap at iteration $k$.

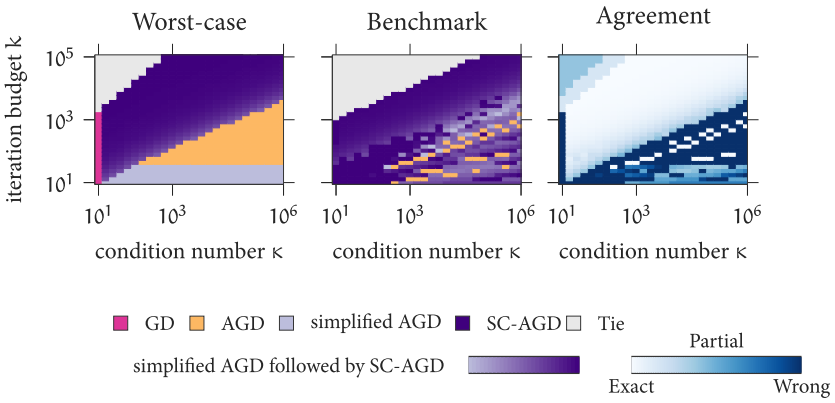



Figure 5.4: Certified recommendation against observed best on diagonal quadratics over a $(\kappa, k)$ grid. Left: the method attaining the smallest certified bound in each $(\kappa, k)$ cell. Middle: the method whose iterates actually attain the smallest gap after $k$ iterations. As the hybrid is parameterized by its switch time, we shade its cells based on the fraction of the budget spent in the S-AGD phase before the switch. Right: the total variation distance between the certified method and the observed best.

**What the grid shows.** Although `Argo.jl`'s recommendations of applicable methods are relevant, it is important to note that the obtained rankings are based on worst-case convergence rates and are therefore not guaranteed to match the observed best method on every problem instance. In this example, we mainly have two discrepancies:

1. For well-conditioned problems, the convergence rate of the optimal-step gradient method is the smallest among the five applicable methods, but it is never the observed best in the benchmark.
2. For ill-conditioned problems and intermediate budgets, `Argo.jl` recommends using the classic accelerated gradient method (AGD), not tailored to strongly convex objectives. In practice, strongly convex variants of AGD or the hybrid method attain a smaller gap.

**Matrix completion over the nuclear-norm ball.** We now focus on nuclear norm constrained matrix completion:

$$\min_{X} \frac{1}{2}\|P_\Omega(X - M)\|_F^2 \quad \text{subject to} \quad \|X\|_* \leqslant r. \tag{5.22}$$

This model estimates a low-rank matrix from a subset of its entries. This has several applications such as image inpainting, collaborative filtering, or recommendation systems. In this example, we consider a rank-5 matrix $M \in \mathbb{R}^{n \times n}$ with $n \in [32, 256]$, and we observe a random subset of its entries with observation rate 0.3. This example is particularly interesting because the cost of an iteration varies across methods. Indeed, conditional gradient methods (or Frank–Wolfe methods) use a linear minimization oracle, which needs only the top singular value and vector whereas projected gradient methods require a full singular value decomposition (SVD) for the projection step. We evaluate `recommendation` over an $(n, \varepsilon)$ grid with 12 values for $n \in [32, 256]$ and 26 accuracies $\varepsilon \in \left[10^{-3}, 10^{-0.5}\right]$.

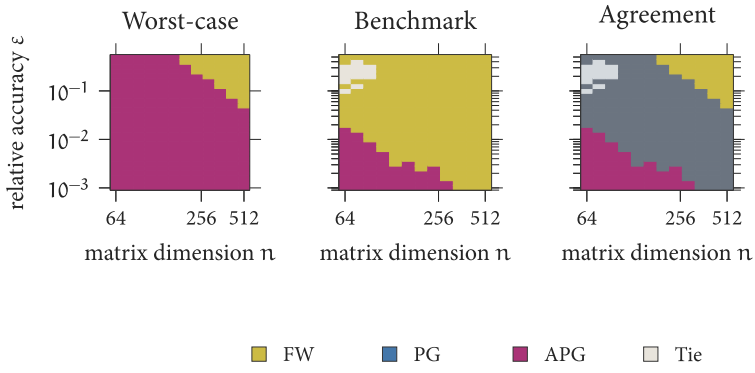



Figure 5.5: Least oracle-expensive method over a $(n, \varepsilon)$ grid. Left: recommendation based on certified iteration bounds and declared oracle costs. Middle: method with the lowest measured oracle work. Right: comparison of the winner sets, with exact agreement in the corresponding method color, overlap in light gray, and disagreement in dark gray.

**What the grid shows.** The boundary between the Frank–Wolfe and APG regions in the worst-case recommendation panel is due to the declared oracle costs and the convergence rates of the two methods. Frank–Wolfe requires $\mathcal{O}\left(n^2/\varepsilon\right)$ oracle work, whereas APG requires $\mathcal{O}\left(n^3/\sqrt{\varepsilon}\right)$, so their crossover scales as $\varepsilon \sim n^{-2}$. APG is therefore selected for small $\varepsilon$, while Frank–Wolfe remains competitive at increasingly small values of $\varepsilon$ as $n$ grows.

The benchmark shows the same dependence but places the crossover at smaller $\varepsilon$, so Frank–Wolfe is preferable over a larger region. The light-gray cells indicate empirical ties between APG and PG; the dark-gray band marks this systematic shift in the Frank–Wolfe crossover. Such discrepancies are inherent to worst-case convergence rates, which hold uniformly over a problem class and are not necessarily tight on a given instance. This appears to be the case here, where the Frank–Wolfe rate is more conservative than the APG one.

## 5.6 APPLICATIONS AND COMPUTATIONAL EXAMPLES

In the previous sections, we have shown how `Argo.jl` goes from a user-provided problem to certificates holding a theorem, a formulation, and a guarantee, and then ranks them under a comparison context. We now show how these certificates can be manipulated further, in three case studies: the composition of two certificates into a guarantee for a hybrid method, the recursive certification of controlled-inexact proximal oracles, and the comparison of methods when the costs of the oracles are unknown. A fourth manipulation, the choice of a free parameter in an exact reformulation, is handled directly by the optimized bound of Section 5.5, which minimizes over such parameters.

### 5.6.1 Switching time for hybrid methods

Consider the following scheme: run a method A for $t$ iterations, then run a method B for the remaining $k - t$ iterations, using the output $x_t$ of A as initial iterate. To obtain a convergence rate for this scheme, we only need the certificate of A to provide, at $x_t$, the initial information required by the theorem of B. When it does, the chained bound is the bound of B evaluated over $k - t$ iterations with the certified guarantee of A as initial information, and it can be smaller than the bound of either method run alone for the full budget.

The certificates make this compatibility a mechanical check. After template applicability, each retained pair records the performance measure it certifies and the initial information it consumes, so the ordered pairs that can be chained are read off directly. Additional pairs become available when the theorem catalogue contains a result that certifies the quantity required as initial information by another theorem. For example, $\mu$-strong convexity of the objective $F$ gives $\frac{\mu}{2}\|x - x^*\|_2^2 \leqslant F(x) - F^*$. A theorem that uses this inequality to certify the squared distance can therefore provide the initial information required by the second phase. At recommendation time, the switching time $t$ is treated as an additional free parameter of the chained

certificate: it enters the optimized bound (5.20) and is chosen by minimizing the chained rate over the integer switching times with a bisection search.

We now work out one such composition by hand. Let $F = f + g$ with $f$ $\mu$-strongly convex and $L$-smooth with a gradient oracle, and $g$ convex with an exact proximal oracle, and fix the same requested guarantee and comparison context as in the guided tour: an upper bound on the last-iterate gap $F(x_k) - F^*$ given the initial information $\|x_0 - x^*\|_2^2 \leqslant R^2$. Two certificates from the theorem catalogue serve as building blocks. The accelerated method SAPG (Taylor et al., 2017c) does not exploit strong convexity and certifies

$$F(x_t) - F^* \leqslant B_{\mathrm{acc}}(t) \stackrel{\mathrm{def}}{=} \frac{2LR^2}{t^2 + 5t + 2}, \tag{5.23}$$

while SC-FISTA (Lin & Xiao, 2014) contracts the Lyapunov function $E(x) = F(x) - F^* + \frac{\mu}{2}\|x - x^*\|_2^2$ by the factor $q \stackrel{\mathrm{def}}{=} 1 - \sqrt{\mu/L}$ at every iteration, and the gap is bounded by this Lyapunov function. The two chain through strong convexity: since $\frac{\mu}{2}\|x_t - x^*\|_2^2 \leqslant F(x_t) - F^*$, the certified gap at the switch gives $E(x_t) \leqslant 2B_{\mathrm{acc}}(t)$, and running SC-FISTA for the remaining iterations yields

$$B_{\mathrm{hyb}}(t) = q^{k-t} \cdot 2B_{\mathrm{acc}}(t), \quad 0 < t < k, \tag{5.24}$$

with the endpoints reproducing the two pure certificates.

Relax $t$ to a real variable on $[0, k]$. The function $\log B_{\mathrm{hyb}}(t)$ is strictly convex on $(0, k)$, and its unique stationary point satisfies $(2t + 5)/\left(t^2 + 5t + 2\right) = -\log q$. When this point lies in $(0, k)$, the integer minimizer is one of the two adjacent integers. For a large condition number $\kappa = L/\mu$, the right side is close to $1/\sqrt{\kappa}$ and the left side behaves like $2/t$, so the minimizer is $t^* \approx 2\sqrt{\kappa}$, independently of the budget $k$ and of $R^2$. The switch thus occurs when the relative per-iteration decrease of the accelerated bound, of order $2/t$, falls below the constant per-iteration decrease $\log\left(\frac{1}{q}\right)$ of the linear phase.

Figure 5.6 shows the three bounds for $\kappa = 10^4$ and $k = 1300$ with $R^2 = 1$. The integer minimizer is $t^* = 197$ (close to the estimate $2\sqrt{\kappa} = 200$). The chained bound guarantees a $1.5 \cdot 10^{-5}$ suboptimality gap whereas the best certificate for one "pure" method only guarantees a $1.1 \cdot 10^{-2}$ suboptimality gap, hence an improvement by a factor of about 700.

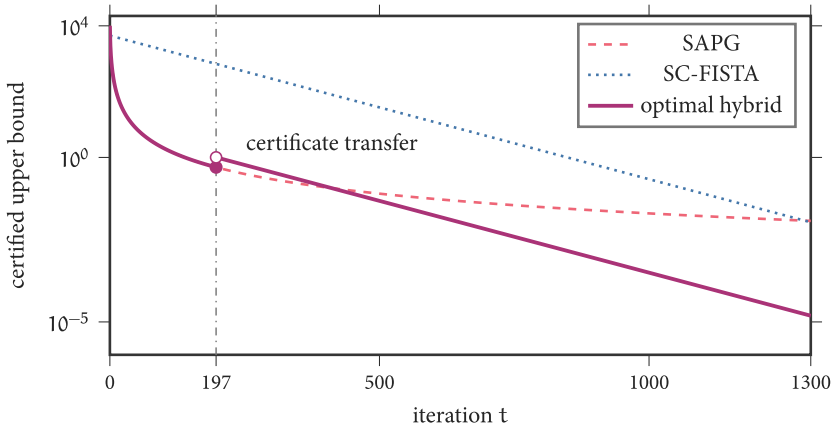



Figure 5.6: Comparison of the SAPG and SC-FISTA bounds with their optimized one-switch composition.

### 5.6.2 Recursive certification of controlled-inexact proximal oracles

Usually, optimization methods which require a proximal oracle assume that it can be computed exactly. Starting with Rockafellar (1976), proximal-point methods were shown to remain convergent under controlled inexactness, classically under summability conditions on the errors. Güler (1992) first examined computational errors in accelerated proximal-point methods; Salzo et al. (2012) later identified a gap in this argument and provided a rigorous analysis recovering the $\mathcal{O}\left(\frac{1}{k^2}\right)$ rate under an appropriate error criterion. For composite problems, Schmidt et al. (2011) and, independently, Villa et al. (2013) established analogous guarantees for basic and accelerated proximal-gradient methods, showing that the exact-method rates are preserved when the proximal errors decay sufficiently fast.

In this subsection, we show how `Argo.jl` can certify applicability of the inexact proximal gradient method in Schmidt et al. (2011), especially in the case where the user does not know how to compute the proximal oracle exactly. The key observation is that each proximal step is itself an optimization problem

$$\min_y g(y) + \frac{\kappa}{2}\|y - z\|_2^2. \tag{5.25}$$

`Argo.jl` can derive assumptions and oracles for that subproblem through the calculus described in Section 5.3.2, and then estimate the number of iterations and oracle work needed to reach the required accuracy by running the certification pipeline recursively on this subproblem.

We call this process *recursive certification* because the oracle required by an outer theorem is justified by certifying a method for the induced proximal subproblem.

We proceed as follows. The comparison context supplies an accuracy schedule $(\varepsilon_j)$ for the controlled-inexact proximal steps. At outer iteration $j$, the returned point must have a subproblem value at most $\varepsilon_j$ above the minimum. Although the center $z_j$ of the proximal subproblem changes at each iteration, every subproblem has the same structure, so `Argo.jl` can enumerate the applicable inner methods once. For each $\varepsilon_j$, it ranks these certificates by oracle work, selects a method $m_j$, and determines the required iteration count $N_{m_j}(\varepsilon_j)$. The selected method may change with $j$ as the requested accuracy $\varepsilon_j$ decreases. In the example below, the inner method uses only the elementary oracles of $g$, so the recursion stops after one level[14].

Let $W_m(\varepsilon)$ denote the certified oracle work required by inner method $m$ to reach a gap $\varepsilon$. At outer iteration $j$, `Argo.jl` selects

$$m_j \in \arg\min_m W_m(\varepsilon_j). \tag{5.26}$$

Inexact proximal gradient also evaluates $\nabla f$ once per outer iteration, so the total oracle work is

$$k c_{\nabla f} + \sum_{j=1}^{k} W_{m_j}(\varepsilon_j). \tag{5.27}$$

If method $m$ evaluates $\nabla g$ once per inner iteration, then $W_m(\varepsilon) = N_m(\varepsilon) c_{\nabla g}$. In the smooth example below, an accelerated-gradient certificate reaches accuracy $\varepsilon$ in $\mathcal{O}(\log(1/\varepsilon))$ gradient calls. For $\varepsilon_j = j^{-p}$, the total oracle work therefore remains $\mathcal{O}(k \log k)$. This total can be compared directly with the oracle work of single-loop methods, that is methods without inner solves.

Figure 5.7 summarizes the recursion.

[14] Increasing the depth only matters if the selected inner method itself requires another unavailable controlled-inexact proximal oracle. In the example below, that does not happen as the inner method uses the gradient or subgradient of $g$. The classical example for inexact proximal methods is when you have TV regularization in inverse problems. As the required proximal operator has no closed-form, it is computed iteratively. Two levels could arise if that inner solver itself requires another unavailable proximal operator, for example, a TV problem with additional intersecting constraints whose projection is computed by another splitting method. However, in practice, such nested structure is usually flattened into a primal-dual splitting method, which are arguably faster and more reliable than such an elaborated scheme.

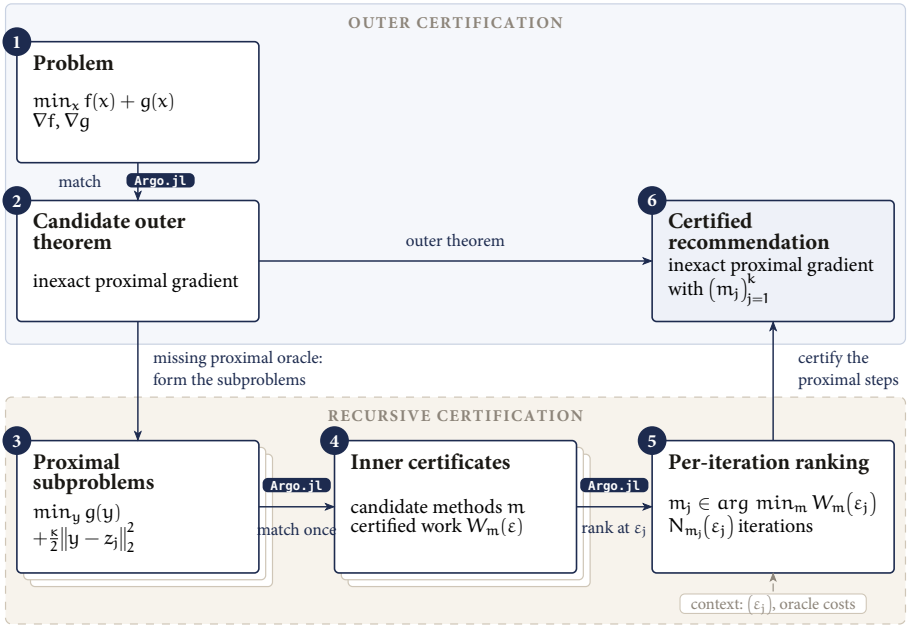



Figure 5.7: Recursive certification with one-time inner matching and per-iteration ranking.

We illustrate the recursion on a small instance. Take $f$ 1-strongly convex and 2-smooth and $g$ convex and $3/2$-smooth, each accessible through value and gradient oracles only. No theorem requiring an exact proximal oracle matches the outer problem, but the inexact proximal gradient template does. With $\kappa = 2$, the proximal subproblem is $7/2$-smooth and 2-strongly convex. The recursive run produces several inner certificates, including gradient and accelerated-gradient methods. `Argo.jl` ranks these certificates separately at each $\varepsilon_j$, so different outer iterations may use different inner methods. If $g$ is instead declared Lipschitz with a subgradient oracle, `Argo.jl` certifies the proximal subproblem with a strongly convex subgradient theorem and retains the Schmidt outer certificate.

### 5.6.3 Comparing methods when oracle costs are unknown

In settings where several oracles are available, it may be hard to quantify the cost of each one relative to the others. This hinders the search for the most computationally efficient method. In such a case, `Argo.jl` can report the number of calls to each oracle for every certificate and only suggest the methods on the Pareto front of oracle calls. Whatever the oracle costs, the method with the least oracle work

necessarily lies on the Pareto front, and the most relevant one for a given setting can be chosen at a glance.

We illustrate this on

$$\min_x f(x) + g_1(x) + g_2(x), \tag{5.28}$$

where the three terms are convex and smooth with $L_f = 1$, $L_{g_1} = 3$ and $L_{g_2} = 1000$, and only their gradients are available. We assume that $\nabla f$ is expensive while $\nabla g_1$ and $\nabla g_2$ are cheap. This happens, for example, when $\nabla f$ involves a costly forward operator, as in MRI, and $g_1$ and $g_2$ are simpler functions such as regularizers. Besides gradient methods on the whole objective, `Argo.jl` returns inexact proximal gradient methods whose proximal steps are certified as in Section 5.6.2. The problem and the ranking for one accuracy schedule are declared as follows.

```
using Argo
using Argo.Recursive

c_f, c_g = symbolic(:cost_f), symbolic(:cost_g)
f = term(:f; properties=[convex(), smooth(1.0)],
    oracles=[oracle(:value), oracle(:gradient; cost=c_f)])
g1 = term(:g1; properties=[convex(), smooth(3.0)],
    oracles=[oracle(:value), oracle(:gradient; cost=c_g)])
g2 = term(:g2; properties=[convex(), smooth(1000.0)],
    oracles=[oracle(:value), oracle(:gradient; cost=c_g)])

# depth=1 also certifies the inner solvers of the inexact
proximal steps
applicable = Recursive.certificates(minimize(f + g1 + g2);
depth=1, reformulation_depth=0)

# accuracy schedule eps_j = c j^(-p) with c = 1e-4 and p = 3
request = RankingRequest(;
    accuracy=1e-2,
    initial_bounds=Dict(:distance_squared => 1.0),
    values=Dict(
        :cost_f => 1.0, :cost_g => 1.0,
        :proximal_accuracy => (j -> 1e-4 * j^(-3)),
        :inner_distance_squared => 1.0,
    ),
    max_iterations=100_000,
)
assessments = rank(applicable, request)
```

Each assessment reports the number of calls to each oracle. The figure repeats this ranking for every schedule and keeps the methods whose call counts are not

dominated. Figure 5.8 shows the calls needed to reach an objective gap of $10^{-2}$ from $\|x_0 - x^*\|^2 \leqslant 1$, for several accuracy schedules of the proximal steps. Three methods stand on the Pareto front:

- the inexact accelerated proximal gradient method with a gradient step on $f$ and a proximal step on $g_1 + g_2$ (blue upward triangles), which needs the fewest calls to $\nabla f$;
- the same method with a gradient step on $f + g_1$ and a proximal step on $g_2$ (purple downward triangles), which needs about twice as many calls to $\nabla f$ and half as many cheap calls;
- the accelerated gradient method on $f + g_1 + g_2$ (green circle), which needs the fewest cheap calls.

They are the best choice when $\nabla f$ costs more than about 750 cheap gradients, between 25 and 750, and below 25, respectively.

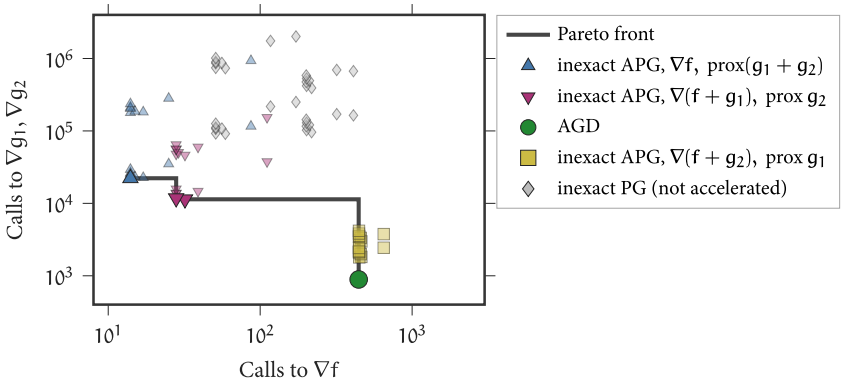



Figure 5.8: Number of evaluations of $\nabla f$ and of $\nabla g_1, \nabla g_2$ needed to reach an objective gap of $10^{-2}$ on $f + g_1 + g_2$, for accuracy schedules $\varepsilon_j = cj^{-p}$ with $c \in \{10^{-2}, ..., 10^{-8}\}$ and $p \in \{3, 5\}$. Transparent markers are dominated variants of the same methods.

## 5.7 Conclusion

### 5.7.1 Summary

This chapter presented `Argo.jl`, a disciplined modeling language for first-order methods. We opened Section 5.1 with the following research question: *Given a structured optimization problem, which templates does it satisfy, possibly after reformulation, which methods are applicable, and how do their comparable worst case*

*guarantees rank?* We propose `Argo.jl` to answer this question algorithmically. Its pipeline is as follows:

1. A problem is written as a structured objective, with a calculus that derives the assumptions, constants, and oracles of each term (Section 5.3),
2. registered reformulation rules may expose equivalent formulations of the same problem (Section 5.3.3),
3. each formulation is matched against the theorem catalogue (Section 5.4) and
4. the certified bounds are ranked under a comparison context (Section 5.5).

An important feature of `Argo.jl` is that, as opposed to benchmarks, each recommendation is a guarantee for the problem at hand. Indeed, we provide a certificate with the formulation on which a method is applied, the theorem that certifies its non-asymptotic convergence rate and why it is applicable as well as the reformulation path that carries this guarantee back to the original problem.

Viewing these certificates as manipulable objects, we show that we can reason beyond the choice of a single method in Section 5.6. In particular, we allow chaining guarantees to build hybrid methods with automatic computation of the switching time, we allow recursive certification of controlled-inexact proximal oracles, and we compare methods whose oracle costs are unknown through the Pareto front of their oracle calls.

### 5.7.2 Limitations and perspectives

`Argo.jl` can't recommend methods beyond what is stored in its theorem catalogue and can't discover formulations that are not reachable by its reformulation rules. Hence, adding a new convergence theorem, defining a new atom with its assumptions and oracles, or adding a reformulation rule enlarges the toolbox for later problems, at once making more methods reachable and more formulations searchable. The value of `Argo.jl` thus compounds as more of the literature is recorded within it.

**Computer-assisted catalogue extension.** Adding atoms and reformulation rules could be partly automated: computer algebra systems could suggest algebraic identities to turn into new reformulation rules, and atoms could be imported from existing libraries of functions with known properties and proximal operators.

**Beyond deterministic first-order methods.** The current theorem catalogue only contains deterministic first-order methods. Covering stochastic methods would require stochastic oracles with explicit noise assumptions, such as bounded variance, and certificates stated in expectation or with high probability; since only certificates with the same performance measure are compared, these would be

ranked separately from deterministic ones. Similarly, second-order methods and general-purpose nonlinear solvers such as Ipopt (Wächter & Biegler, 2006) would require Hessian oracles and functional constraints in the modeling language. As such solvers usually come with asymptotic or local guarantees only, `Argo.jl` could certify that they apply but could not rank them by worst-case bound, except for interior-point methods based on self-concordant barriers, whose worst-case iteration bounds (Nesterov, 2004) can be compared with first-order methods through their oracle work. Such certificates would remain useful, since there is often little guidance on which solver to use for a given problem or how to set its parameters, and a recommendation backed by a convergence theorem, even a conservative one, gives a principled starting point. Solvers with local guarantees could moreover enter the hybrid schemes of Section 5.6.1, as a second phase started once a first-order method certifies the initial information they require, such as a bound on the distance to a solution.

# 6 FAIRNESS-INFORMED PARETO OPTIMIZATION


## ABSTRACT

Despite their promise, fair machine learning methods often yield Pareto-inefficient models, in which the performance of certain groups can be improved without degrading that of others. This issue arises frequently in traditional in-processing approaches such as fairness-through-regularization. In contrast, existing Pareto-efficient approaches are biased towards a certain perspective on fairness and fail to adapt to the broad range of fairness metrics studied in the literature. In this chapter, we present BADR, a simple framework to recover the optimal Pareto-efficient model for any fairness metric. Our framework recovers its models through a Bilevel Adaptive Rescalarization procedure. The lower level is a weighted empirical risk minimization task where the weights are a convex combination of the groups, while the upper level optimizes the chosen fairness objective. We equip our framework with two novel large-scale, single-loop algorithms, BADR-GD and BADR-SGD, and establish their convergence guarantees. We release `badr`[15], an open-source Python toolbox implementing our framework for a variety of learning tasks and fairness metrics. Finally, we conduct extensive numerical experiments demonstrating the advantages of BADR over existing Pareto-efficient approaches to fairness.


## BASED ON

*This chapter stems from research conducted under the supervision of Professor Yassine Laguel, with the collaboration of Professor Samuel Vaiter.*
*It is based on the following preprint: Tanji, S., Vaiter, S., & Laguel, Y. (2026). Fairness-informed Pareto Optimization: An Efficient Bilevel Framework.* Arxiv Preprint Arxiv:2601.13448.*, submitted to Journal of Machine Learning Research (JMLR) in January 2026.*

## IN THIS CHAPTER

[15] https://github.com/SofianeTanji/PhD-Software/Badr

Section 6.2 motivates multi-objective approaches to fairness in machine learning and introduces BADR, our bilevel framework for computing optimal Pareto-efficient models with respect to a given fairness metric. Section 6.3 presents scalable algorithms for solving the resulting bilevel optimization problem, along with their convergence analysis. In Section 6.4, we introduce `badr`, our open-source Python package, which supports efficient implementation of our approach and evaluation of its numerical performance on several standard learning tasks. Section 6.5 is devoted to extensive empirical studies evaluating the performance of our method against state-of-the-art approaches to Pareto-efficient fair learning. Section 6.6 closes the chapter with a discussion of possible future work.

## 6.1 INTRODUCTION

Data-driven decision systems are now commonly used in high-stakes domains such as employment, education, law enforcement, and health care (Citron & Pasquale, 2014).

Their widespread deployment in systems that assist with, and sometimes replace, human decision-making has been accompanied by increased concern about unfairness and bias based on sensitive attributes such as gender, race, or religion (Cross et al., 2024; Mehrabi et al., 2021; Obermeyer et al., 2019). Motivated by evidence that learned models can perform unevenly across sensitive groups (Mehrabi et al., 2021), research on algorithmic fairness has proposed a range of formulations and metrics (Berk et al., 2017; Feldman et al., 2015; Hardt et al., 2016) to quantify and alleviate the bias of standard learning algorithms. Yet optimizing these criteria typically comes at a cost in predictive performance, inducing an inherent fairness–accuracy trade-off (Menon & Williamson, 2018). This tension is particularly pronounced for in-processing methods that incorporate constraints or penalties into the learning algorithm (Do et al., 2022), often yielding models that are *inefficient* from a performance standpoint. Simply put, it is often possible to improve the performance of one group without harming others, indicating that the model is not operating on the *Pareto front* of group performance.

This limitation has spurred interest in group fairness formulations (Dwork et al., 2012) that consider alternative aggregations of group performance to the standard empirical risk minimization (ERM) objective. For instance, the minimax group fairness framework (Martinez et al., 2020) minimizes the worst-group loss, ensuring that no group's performance can be improved without degrading another's. More broadly, fairness risk measures such as the Conditional Value at Risk (CVaR) (Rockafellar & Uryasev, 2000) interpolate between the robust minimax approach and the expectation-based ERM loss. While these methods guarantee Pareto efficiency, as detailed in Section 6.2.3, their impact on specific fairness metrics remains unpredictable and uncontrollable.

In this chapter, we introduce BADR, a framework for selecting, among Pareto-efficient models, the model that minimizes a user-specified fairness metric. We cast the task as a bilevel formulation: the lower level enforces Pareto efficiency across groups, and the upper level selects the point that minimizes the chosen fairness metric. We propose BADR-GD, a large-scale single-loop stochastic algorithm designed specifically for this bilevel problem, and prove convergence under standard assumptions. We release an open-source Python package that implements our approach for several fairness metrics. Experiments on real-world datasets

show improvements over existing methods on the targeted fairness metric while preserving Pareto efficiency across groups.

## Related work

**Multi-objective optimization.** In this work, we tackle fair machine learning within a multi-objective optimization framework. That is, group performance is measured by the corresponding group losses, and success is assessed through Pareto efficiency (Ehrgott, 2005). The set of Pareto-efficient models, called the Pareto front, can be searched through different methods, including scalarization methods (Ehrgott, 2005; Eichfelder, 2008), the $\epsilon$-constraint method (Mavrotas, 2009), and particle swarm optimization (Coello et al., 2004). Here, we build on a standard scalarization method, the weighted-sum approach (Geoffrion, 1968; Zadeh, 1963), which replaces the multi-objective formulation with an arbitrary convex combination of the objectives. This approach is widely used in practice and is guaranteed under mild assumptions to output Pareto-efficient solutions (Ehrgott, 2005; Geoffrion, 1968). Yet the interplay between the convex weights and the quality of the resulting solution remains poorly understood in general (Das & Dennis, 1997; Ehrgott, 2005). Some methods attempt to explore the entire Pareto front by varying the weights (Kim & Weck, 2004; Ryu & Min, 2019), while other methods compare weights based on their relative impact on the objective (Ayan et al., 2023). In this work, we present a novel data-driven approach to weight selection in the weighted-sum method. Our method optimizes a user-specified fairness metric over the Pareto front of group performance, thereby tailoring the solution to the stakeholders' fairness concerns. Beyond fair learning, our approach is applicable to general multi-objective problems whenever a utility function is given to measure the quality of Pareto-efficient solutions.

**Efficient fair learning.** To mitigate unfairness, three families of interventions are common: pre-processing (Calmon et al., 2017; Lahoti et al., 2019; Ruoss et al., 2020; Xu & Strohmer, 2023; Zemel et al., 2013), which modifies the data or labels before training (e.g., through reweighting, resampling, or learning fair representations); in-processing (see, e.g., Zafar, Valera, Gomez Rodriguez, et al., 2017; Zafar, Valera, Gomez-Rodriguez, et al., 2019; Donini et al., 2018; Cruz et al., 2022; Cotter et al., 2019; Celis et al., 2019), which alters the training process itself to promote fairness in the learned model; and post-processing (Lohia et al., 2019), which adjusts the scores or decision thresholds of a trained model to satisfy a target criterion. Within in-processing, arguably the most common approach is to incorporate fairness requirements in the optimization problem with either explicit fairness constraints or penalization terms (Do et al., 2022). By formulating the initial optimization

problem to penalize violations of a chosen fairness criterion (e.g., adding a constraint on demographic parity or a penalty term for equalized odds), the model is encouraged to be fairer during learning. However, from a Pareto perspective this approach cannot guarantee efficient solutions; see our discussion in Section 6.2: a model trained with such penalties may still be improvable for some groups without harming others.

Most Pareto-efficient methods in the fairness literature rely on data reweighting and have been cast as both pre- and in-processing techniques. Regarding pre-processing methods, for instance, Sivarajkumar et al. (2023) propose reweighting data by group with weights inversely proportional to group frequencies. We also note that several works consider reweighting jointly per group and class—e.g., Kamiran & Calders (2012)—to reduce certain fairness metrics such as demographic parity. Yet allowing weights to depend on class takes the method out of our multi-objective framework: the resulting weighted loss is no longer a convex combination of group losses, and Pareto efficiency is not guaranteed. Regarding in-processing methods, in the same vein as Kamiran & Calders (2012), some training procedures dynamically estimate weights jointly on group and class to mitigate certain fairness metrics (Jiang & Nachum, 2019; Krasanakis et al., 2018). Alternatively, several methods have considered adversarial reweighting schemes to promote fairness during training. This line of work, designated as minimax fairness (Abernethy et al., 2020; Diana et al., 2021; Hashimoto et al., 2018; Martinez et al., 2020), has expanded to include distributionally robust optimization approaches (Pillutla et al., 2024; Tsang & Shehadeh, 2025; Williamson & Menon, 2019), and has the benefit of systematically yielding Pareto-efficient models; see Section 6.2.3 for details. Yet, given a specific fairness metric, it is difficult to guarantee the quality of these robust models.

Finally, we note that several works have explored bilevel optimization to dynamically adjust groupwise weights. For instance, Ozdayi et al. (2021) and Roh et al. (2020) consider a bilevel formulation where the upper level minimizes arbitrary fairness metrics but the lower level allows for weights to depend on both group and class, thereby losing Pareto efficiency. Alternatively, Martinez et al. (2020), Abernethy et al. (2020), and Shekhar et al. (2021) propose a bilevel framework where the lower level minimizes a weighted group loss, and the upper level optimizes the worst-case group loss. Furthermore, some bilevel methods, such as those proposed in Kamani et al. (2021), have been developed to address multi-objective problems that aim to balance different fairness metrics and accuracy. These approaches lead to formulations and considerations that differ from ours: competing loss functions do not share the same regularity properties and are typically supported by weaker

convergence guarantees. Finally, we note that none of these papers provide large-scale optimization methods with convergence guarantees that effectively solve their bilevel formulations.

**Bilevel optimization.** Bilevel optimization (Dempe, 2002; Dempe & Zemkoho, 2020) plays an increasing role in machine learning applications, including hyper-parameter optimization (Bertrand et al., 2022), meta-learning (Chayti & Jaggi, 2024), and fair machine learning, as discussed above. Among existing approaches for bilevel learning problems, we note two main categories: penalization-based approaches (Kwon et al., 2023) that transform the bilevel problem into a single-level one by adding a penalty term to the upper-level objective, and implicit methods (Domke, 2012; Maclaurin et al., 2015) that approximate hypergradients through implicit differentiation or unrolling techniques. Early deterministic works relied on double-loop algorithms, where the inner loop iteratively solved the lower-level problem to near-optimality (Ghadimi & Wang, 2018; Gong et al., 2024; Pedregosa, 2016). More recent studies have proposed single-loop schemes that update both levels simultaneously (Chen et al., 2022; Dagréou et al., 2022), thereby avoiding the need to tune the number of inner iterations. To enable implicit differentiation and theoretical guarantees, these methods usually assume the lower-level problem is smooth (with Lipschitz-continuous Hessians) and strongly convex (Dagréou et al., 2022; Ghadimi & Wang, 2018; Pedregosa, 2016), or at least satisfies the Polyak–Łojasiewicz condition (Ji et al., 2021). Yet, in our setting, the multiplicative interaction between model weights and group-level losses induces unbounded cross derivatives, violating the smoothness and convexity assumptions commonly required in the analysis.

### Contributions

We have four main contributions.

**badr.** We introduce BADR for fair learning, which trains models to minimize a user-specified fairness metric subject to groupwise Pareto efficiency. BADR treats groupwise empirical losses as multiple objectives, seeking solutions where no group loss can be decreased without increasing at least one other. Within this efficient set, BADR minimizes the selected fairness metric. The key idea is to learn the scalarization parameters (group weights) via a bilevel optimization formulation. The lower level enforces efficiency, whereas the upper level optimizes the fairness criterion, rather than fixing the weights a priori. This perspective provides a unified basis to compare Pareto-efficient baselines for fair learning (e.g., balanced reweighting and minimax fairness) and enables metric-driven improvements in the learned model.

**Large-scale bilevel optimization methods.** Our second contribution is the development of two single-loop bilevel algorithms, BADR-GD and BADR-SGD, tailored to BADR in the deterministic and stochastic settings, respectively. These algorithms jointly update the model parameters, the dual variable associated with the lower-level optimality conditions, and the group-weight vector through coupled gradient steps. The proposed updates require only first-order information on the fairness metric and approximate second-order information on group losses. In contrast to traditional bilevel approaches that rely on costly inner-outer loops or strong smoothness assumptions, BADR-GD and BADR-SGD operate under mild regularity conditions. We establish theoretical convergence guarantees in this general bilevel setting, achieving complexity rates of $\mathcal{O}(1/T)$ in the deterministic case and $\mathcal{O}\left(1/\sqrt{T}\right)$ in the stochastic case. Notably, our convergence metric jointly captures stationarity of the implicit upper-level objective and suboptimality of the lower-level solution. A central challenge in our analysis arises from the absence of *global* smoothness of the lower-level objective with respect to the upper-level variable. In the deterministic setting, we overcome this difficulty by deriving a novel uniform bound on the dual variable generated by the algorithm. In the stochastic setting, such a uniform bound is no longer attainable. To address this issue, we introduce two novel technical arguments: (i) a clipping mechanism that controls the influence of the dual variable in the estimation of the implicit function, and (ii) a new Lyapunov function that extends those previously considered in the literature (Dagréou et al., 2022) by incorporating an additional term specifically designed to handle the unboundedness of the dual variable. These results constitute a contribution to the bilevel optimization literature beyond the fairness application of this chapter.

**Open-source Python package.** We release `badr`[16], an open-source Python package that implements BADR. `badr` conforms to the `scikit-learn` interface, which makes the training and analysis of fair models straightforward. The package ships with seven unfairness metrics and supports user-defined custom metrics. Eleven widely used tabular datasets can be readily fetched, and users can also supply `pandas` DataFrames, removing the need to write custom data loaders. The package directly integrates existing learning models such as linear regression, logistic regression, and support vector machines. Finally, we provide abstractions for the oracles, allowing any user to plug in their own optimization algorithm to solve the underlying problem. We offer an in-depth empirical comparison of several optimization methods (including performance profiles and scalability experiments), highlighting the effectiveness of our proposed algorithms BADR-GD and BADR-SGD.

[16] Throughout this chapter, `badr` (lowercase) denotes the Python package we release, whereas BADR refers to our overall framework.

**Extensive numerical experiments.** Using the `badr` package, we conduct extensive numerical experiments on real-world datasets and seven fairness metrics. We provide numerical evidence that BADR improves the targeted fairness metric over other Pareto-efficient baselines, namely uniform sampling, balanced sampling, minimax fairness, and one-group fitting. We visualize how the unfairness metrics behave along the Pareto front for two and three groups, thereby supporting the need to optimize over the Pareto front. We also compare BADR with these baselines for larger numbers of groups and again show that it improves the targeted fairness metric without either (1) compromising Pareto efficiency with respect to group performance or (2) degrading accuracy (for classification) or RMSE (for regression) on the test set.

## 6.2 FAIRNESS THROUGH A PARETO LENS

We consider a general prediction problem in which the data consist of triplets $(x_i, y_i, a_i)$. Specifically, $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$ represents the input-output pair for individual $i$, and $a_i \in \mathcal{A} = \{1, ..., S\}$ indicates their sensitive attribute (e.g., gender, race, or similar characteristics). Traditionally, predictors $f(w, \cdot) : \mathcal{X} \to \mathcal{Y}$ are estimated through the empirical risk minimization (ERM) principle. Given training data $(x_i, y_i, a_i)_{i \in \{1,...,n\}}$, this entails solving

$$\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} [\ell(f(w, x_i), y_i)], \tag{6.1}$$

where $w \in \mathbb{R}^d$ parametrizes the predictor. Notably, ERM disregards individuals' sensitive attributes, potentially resulting in biases across the communities involved (Blodgett et al., 2016). In this work, we seek to develop a novel learning method that reduces the disparity in performance across communities. Our approach aligns with the framework of in-processing fairness methods (Caton & Haas, 2024). In this section, we present an adaptive reweighting scheme to enhance fairness while guaranteeing Pareto efficiency with respect to group performance.

### 6.2.1 Inefficiency of fairness by regularization approaches

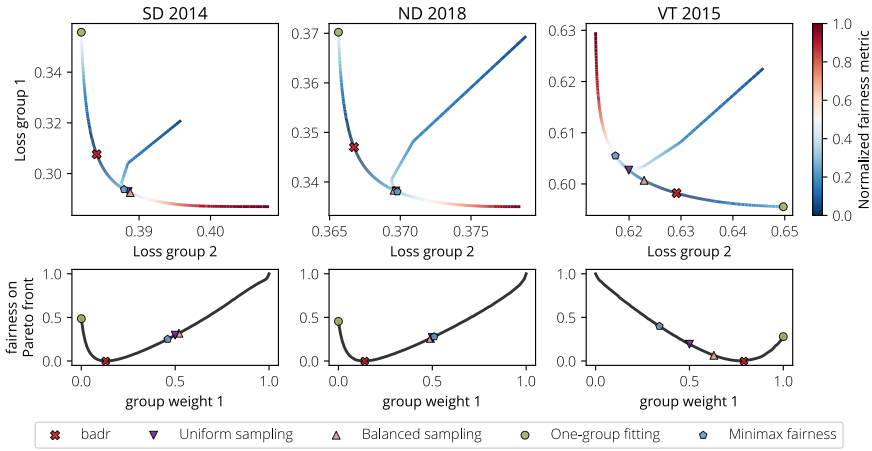



Figure 6.1: Illustration of the impact of fairness regularization on Pareto efficiency for a logistic regression task evaluated on three subsamples of the ACSEmployment dataset (Ding et al., 2021). The first row reports the groupwise losses for the two sensitive groups (male and female). The two curves correspond to the Pareto front and the regularization path induced by the fairness-by-penalization approach, respectively. The second row reports the values of the considered fairness metric (individual fairness, Berk et al., 2017) on the Pareto front, together with the performance of several Pareto-efficient fairness methods compared with BADR.

A natural starting point for quantifying performance disparities across groups is to explicitly examine their respective losses. Specifically, for each sensitive attribute $a \in \mathcal{A}$, we define the *unregularized group loss* $F_a : w \mapsto F_a(w) = n_a^{-1} \sum_{i:a_i=a} \ell(f(w, x_i), y_i)$, where $n_a = \#\{i : a_i = a\}$ denotes the number of data points corresponding to the sensitive attribute $a$. Group losses allow us to extend the original empirical risk minimization problem (6.1) into a multi-objective formulation:

$$\min_{w \in \mathbb{R}^d} [F_1(w), ..., F_S(w)]. \tag{6.2}$$

Considering fairness through group losses underpins many fairness-oriented learning methods. A common strategy, often termed minimax fairness, reallocates sampling weight toward the worst-performing groups, collapsing (6.2) into a single-objective problem (see Section 6.2.3). Nonetheless, the full multi-objective formulation in (6.2) admits a broader spectrum of solutions, offering greater flexibility in pursuing fairness. Such solutions are typically characterized by the concept of Pareto efficiency:

**Definition 6.1.** (Pareto Efficiency) A model $\bar{w}$ is called *Pareto-efficient* for the group losses $(F_a)_{a\in\mathcal{A}}$ if, for all $w \in \mathbb{R}^d$, one of the following two conditions holds:

$$\begin{aligned} &\text{(i)} \quad F_a(w) \geqslant F_a(\bar{w}), \quad \text{for all} \quad a \in \mathcal{A}, \\ &\text{(ii)} \quad F_a(w) > F_a(\bar{w}), \quad \text{for some} \quad a \in \mathcal{A}. \end{aligned} \tag{6.3}$$

The set of Pareto-efficient points for (6.2), subsequently denoted by $\mathcal{P}$, is called the *Pareto front*.

Intuitively, Pareto-efficient solutions characterize models for which performance cannot be improved for any group without adversely affecting at least one other. While this requirement seems natural, it is not necessarily achieved by current in-processing fairness methods. This can be observed, for instance, in fairness-by-constraint methods, which typically involve augmenting the empirical risk minimization (ERM) problem (6.1) with a fairness constraint,

$$\min_{w\in\mathbb{R}^d} \quad \frac{1}{n}\sum_{i=1}^{n} \ell(f(w, x_i), y_i) \text{ s.t. } \mathcal{U}(w) \leqslant \epsilon, \tag{6.4}$$

or its relaxation through penalization (Do et al., 2022),

$$\min_{w\in\mathbb{R}^d} \quad \frac{1}{n}\sum_{i=1}^{n} \ell(f(w, x_i), y_i) + \frac{1}{\nu}\,\mathcal{U}(w). \tag{6.5}$$

Here, $\epsilon > 0$ (respectively, $\nu > 0$) is a user-defined tolerance level that governs the permissible degree of unfairness, and $\mathcal{U} : \mathbb{R}^d \to \mathbb{R}$ denotes the fairness metric used to quantify model bias—we review a number of standard fairness metrics in Section 6.A.

In Figure 6.1, we illustrate the effect of fairness regularization on Pareto efficiency across three classification tasks. We train a logistic regression model on three distinct subsamples of the ACSEmployment dataset (Ding et al., 2021). In each case, gender is the sensitive attribute, with male and female as the two groups. The model is augmented with an individual-fairness penalty (Berk et al., 2017), and we vary the penalty weight $\nu$ over a logarithmic grid from $10^{-6}$ to $10^{-0.5}$. The top row plots the Pareto front of group losses and the regularization path starting from uniform weights. Both are colored by the individual fairness metric, with the color scale shown by the color bar. In the bottom row, we show how the fairness metric changes with the group weights. This view makes changes in the metric more apparent than the color map. Across all three datasets, the resulting trajectories significantly depart from the Pareto front represented by the subgroup losses (in blue). This demonstrates that naive penalization produces solutions that are dominated and

therefore inefficient from a Pareto perspective. We therefore first restrict our search to the Pareto front. If fairness must later be improved beyond Pareto efficiency, it is more natural to penalize an optimally reweighted loss than the empirical risk.

### 6.2.2 Scalarization and the weighted-sum method

Problem (6.2) may be addressed by a broad range of optimization approaches aiming to recover its Pareto front. These include scalarization methods, a priori methods, and no-preference methods—for general references, see Branke (2008) and Marler & Arora (2004). First, we denote the probability simplex in $\mathbb{R}^S$ by $\Delta_S$.

$$\Delta_S \stackrel{\text{def}}{=} \left\{ \lambda \in \mathbb{R}^S_+ : \sum_{a=1}^{S} \lambda_a = 1 \right\}. \tag{6.6}$$

In this work, we build on a standard scalarization strategy in the multiobjective literature, the weighted-sum method (Gass & Saaty, 1955; Geoffrion, 1968), which, given a distribution of weights $\lambda \in \Delta_S$, turns (6.2) into the single optimization problem:

$$\min_{w \in \mathbb{R}^d} \sum_{a \in \mathcal{A}} \lambda_a F_a(w) \tag{6.7}$$

Under mild conditions, solutions of (6.7) are guaranteed to be Pareto-efficient. Their closure also spans the entire Pareto front when the group losses are strongly convex.

**Proposition 6.2.** (See Geoffrion, 1968) *Let $\lambda \in \mathrm{ri}(\Delta_S)$ be fixed. Any solution $w_\lambda$ of (6.7) is Pareto-efficient for the multi-objective problem (6.2). Conversely, if the group losses $F_a$ are convex and at least one of them is strictly convex, then any Pareto-efficient point for (6.2) belongs to the closure of $\{w_\lambda, \lambda \in \Delta_S\}$.*

When the learning weights $\lambda_a$ correspond to the proportion $n_a/n$ of each group in the training sample, one recovers the standard ERM problem (6.1), which, by the preceding proposition, yields a Pareto-efficient model. Alternatively, the fairness literature has considered a broad family of data reweighting strategies to alleviate the unfairness of ERM's output. Yet these approaches are mostly restricted to binary classification tasks—e.g., Calders et al. (2009) and Krasanakis et al. (2018) —and further require weights to jointly depend on the sensitive attribute $a$ and the output label $y$. This is the case, for instance, in Calders et al. (2009), which adopts an importance sampling approach to achieve independence in the training data between the sensitive attribute and the output variable. As a consequence, group loss performance may no longer exhibit Pareto efficiency.

### 6.2.3 Minimax fairness and fairness risk measures as robust scalarizations

From a group-loss perspective, minimax fairness (Abernethy et al., 2020; Diana et al., 2021; Hashimoto et al., 2018; Martinez et al., 2020) constitutes a popular approach to fairness that focuses on improving performance for the worst-performing groups

$$\min_{w} \max_{a \in \mathcal{A}} F_a(w). \tag{6.8}$$

Solutions of (6.8) offer robust guarantees on the output model $w$ by minimizing the worst-case group error, often resulting in a reduced performance gap across subgroups. This highlights their advantage over equal error rate approaches (Diana et al., 2021), which may enforce parity by uniformly degrading performance, even at the expense of disadvantaged groups.

Such solutions also offer computational advantages over other in-processing methods because they can draw on the extensive literature on stochastic minimax optimization (Beznosikov et al., 2023). In particular, optimization algorithms for (6.8) are frequently supported with theoretical guarantees including the preservation of convexity for linear tasks (logistic or linear regression) and convergence rates toward an optimal or critical point in the nonconvex setting. These properties have spurred interest in minimax fairness approaches across a broad span of learning tasks, notably in federated learning (Mohri et al., 2019).

A common limitation of the formulation (6.8) is its potential focus on the performance of a single group, which may be inadequate in situations where many groups are involved. To this end, extensions of (6.8) have been considered to provide robustness to distributions over groups, leading to the minimax formulation

$$\min_{w} \max_{Q \in \mathcal{D}} \mathbb{E}_{a \sim Q}[F_a(w)]. \tag{6.9}$$

where $\mathcal{D} \subseteq \Delta_S$ denotes the ambiguity set of probability distributions over the $S$ sensitive groups considered by the inner worst-case problem. Several studies have analyzed these extensions under different formulations of the ambiguity set (Pillutla et al., 2024) and established connections with the theory of risk measures (Tsang & Shehadeh, 2022; Williamson & Menon, 2019).

It turns out that problem (6.8) and its extension (6.9) both output Pareto-efficient models when group losses are convex:

**Proposition 6.3.** *Assume the group losses* $(F_a)_{a \in \mathcal{A}}$ *are convex. Then any solution of (6.8) or (6.9) is weakly Pareto-efficient.*

Proposition 6.3 underpins the motivation in Martinez et al. (2020) for minimax fairness, although the formulation presented there remains expressed as $\min_{w\in\mathcal{P}} \max_{a\in\mathcal{A}} F_a(w)$ with an explicit restriction to $\mathcal{P}$. Subsequent works (Abernethy et al., 2020; Diana et al., 2021) have relaxed the Pareto-front constraint; this relaxation follows directly from Proposition 6.3. A proof of Proposition 6.3 is provided in (Sagawa et al., 2019, Proposition 1) for the analogous setting in which the ambiguity set $\mathcal{D}$ consists of distributions supported on the training set, rather than on the set of sensitive attributes.

### 6.2.4 BADR: an optimal Pareto-efficient approach to group fairness

In this chapter, we propose to learn optimal Pareto-efficient models for (6.2) for an arbitrary measure of unfairness $\mathcal{U} : \mathbb{R}^d \to \mathbb{R}$. Using the implicit characterization in (6.7), we cast this task as a bilevel program, which we term BADR (Bilevel Adaptive Rescalarization), and formulate as follows:

$$\begin{aligned} \min_{\lambda\in\Delta_S} \quad & \mathcal{U}(w_\lambda) \\ \text{s.t.} \quad & w_\lambda \in \arg\min_{w\in\mathbb{R}^d} \sum_{a\in\mathcal{A}} \lambda_a F_a(w) \end{aligned} \tag{6.10}$$

When the group losses $F_a$ are strongly convex, solutions of (6.10) are guaranteed to be Pareto-efficient. If the $F_a$ are moreover twice continuously differentiable, the implicit function theorem (Dontchev & Rockafellar, 2014) ensures that $\lambda \mapsto w_\lambda$ is smooth, so (6.10) reduces to the generally nonconvex single-level problem $\min_{\lambda\in\Delta_S} \mathcal{U}(w_\lambda)$. Furthermore, provided that the unfairness metric $\mathcal{U}$ is differentiable or admits a tractable smooth approximation, problem (6.10) is amenable to standard bilevel programming methods for large-scale learning (Dagréou et al., 2022; Pedregosa, 2016). In particular, for any initial point $\lambda$ in the simplex, the unfairness of the associated model $w_\lambda$ can be decreased until a critical point of the implicit loss $\lambda \mapsto \mathcal{U}(w_\lambda)$ is reached.

Several bilevel approaches to in-processing fairness have been explored in the literature. Roh et al. (2020) propose learning a sampling distribution depending jointly on sensitive attributes and labels. In particular, the lower-level objective may not correspond to a weighted sum of group losses, which can compromise the Pareto efficiency of the output model. Martinez et al. (2020) and Shekhar et al. (2021) study the particular case of (6.10) in which the unfairness metric $\mathcal{U}$ corresponds to maximum group loss. Ozdayi et al. (2021) study a problem similar to (6.10), where the upper level additionally includes the population loss $w \mapsto \sum_{a\in\mathcal{A}} \frac{n_a}{n} F_a(w)$ as a penalty term. While several fairness metrics are considered in that work, no connection to Pareto efficiency is established. Multi-objective

approaches for group fairness have also been considered in Liu & Vicente (2022), where the group losses and fairness metrics are taken as competing objectives.

Our general formulation (6.10) offers two key advantages over existing approaches in the literature. To begin with, it is the first formulation that can accommodate a broad class of fairness metrics while simultaneously ensuring Pareto efficiency across groups. Second, as shown in the following section, it enables efficient optimization with provable guarantees.

## 6.3 LARGE-SCALE METHODS

In this section, we introduce BADR-GD, a single-loop method for Bilevel ADaptive Rescalarization, to solve the bilevel program (6.10). Section 6.3.1 presents the algorithmic details of BADR-GD. Section 6.3.2 establishes its convergence properties under standard assumptions in the bilevel optimization literature. This method complements the algorithmic methods we developed for small- to medium-scale problems in Section 6.4.

### 6.3.1 Algorithms

Algorithm 6.1 presents BADR-GD, a simple method to solve the bilevel problem (6.10) for large-scale instances. BADR-GD iteratively updates three variables: the learning model $w$, the dual variable $v$ associated with the optimality condition of the lower-level problem in (6.10), and the fairness reweighting vector $\lambda$. To state the second-order steps, we denote the Hessian of a scalar function $G$ by $\nabla^2 G$; subscripts indicate the differentiated variables for functions of several arguments. Specifically, at each iteration $t$, it performs a gradient-descent step on the reweighted objective $w \mapsto \sum_{a\in\mathcal{A}} \lambda_{t,a} F_a(w)$ (line 3) and a gradient step on the quadratic objective $v \mapsto \frac{1}{2} v^\top Q_t v + \nabla \mathcal{U}(w_t)^\top v$, where $Q_t$ denotes the weighted Hessian $Q_t \triangleq \sum_{a\in\mathcal{A}} \lambda_{t,a} \nabla^2 F_a(w_t)$ (line 4). Finally, the fairness weights $\lambda$ are updated via an approximate projected gradient step on the objective $\lambda \mapsto \mathcal{U}(w^*(\lambda))$ (line 5).

**Algorithm 6.1.** (BADR-GD)

1 **Inputs:** Starting points $w_0, v_0, \lambda_0$; step sizes $\tau, \rho, \gamma$

2 **for** $t = 0$ **to** $T - 1$:

3 $w_{t+1} \leftarrow w_t - \tau \nabla \mathbf{F}(w_t)^\top \lambda_t$ // $\nabla \mathbf{F}(w_t) \in \mathbb{R}^{s \times d}$ with $a^{\text{th}}$ row equal to $\nabla F_a(w_t)$

4 $v_{t+1} \leftarrow v_t - \rho\big(\nabla \mathcal{U}(w_t) + \sum_{a \in \mathcal{A}} \lambda_{t,a} \nabla^2 F_a(w_t) v_t\big)$

5 $\lambda_{t+1} \leftarrow \mathrm{Proj}_{\Delta_s}(\lambda_t - \gamma \nabla \mathbf{F}(w_t) v_t)$

6 **return** $w_T, \lambda_T$

A notable advantage of this method is its simplicity, as it only requires (approximate) second-order information about the group losses $F_a$ and first-order information on the fairness metric $\mathcal{U}$. Furthermore, its single-loop nature makes it particularly well-suited for large-scale problems, as it avoids the costly inner-loop optimization of the lower-level problem in (6.10).

Still, exact gradients for the group losses $F_a$ or the fairness metric $\mathcal{U}$ are rarely available in typical learning scenarios. We therefore extend our method in Algorithm 6.2 to the case where only stochastic estimates of these gradients are available. Specifically, we assume access to stochastic estimates $\tilde{\nabla}\mathcal{U}(w)$, $\tilde{\nabla} F_a(w)$, and $\tilde{\nabla}^2 F_a(w)$ of $\nabla \mathcal{U}(w)$, $\nabla F_a(w)$, and $\nabla^2 F_a(w)$, respectively, which may be computed through minibatch samples. A notable difference from BADR-GD is the use of a clipping operator on the gradient estimate for the upper variable $\lambda$ (line 5 of Algorithm 6.2), which is instrumental in the derivation of our convergence results in Section 6.3.2.

**Algorithm 6.2.** (BADR-SGD)

**Inputs:** Starting points $w_0, v_0, \lambda_0$;
step sizes $\tau, \rho, \gamma$; clipping threshold $C_\gamma$
**for** $t = 0$ **to** $T-1$:
  $w_{t+1} \leftarrow w_t - \tau \tilde{\nabla} \mathbf{F}(w_t)^\top \lambda_t$ // $\tilde{\nabla}\mathbf{F}(w_t) \in \mathbb{R}^{s \times d}$ with $a^{\text{th}}$ row equal to $\tilde{\nabla} F_a(w_t)$
  $v_{t+1} \leftarrow v_t - \rho\Big(\tilde{\nabla}\mathcal{U}(w_t) + \sum_{a \in \mathcal{A}} \lambda_{t,a} \tilde{\nabla}^2 F_a(w_t) v_t\Big)$
  $\lambda_{t+1} \leftarrow \text{Proj}_{\Delta_s}\Big(\lambda_t - \gamma\, \mathbf{Clip}\Big(\tilde{\nabla}\mathbf{F}(w_t) v_t\Big)\Big)$
**return** $w_T, \lambda_T$

### 6.3.2 Convergence analysis

To analyze the bilevel formulation (6.10), we rewrite it in the form of a general bilevel optimization problem:

$$\begin{aligned} \min_{\lambda \in \mathcal{C}} \quad & \mathcal{U}(w^*(\lambda), \lambda) \\ \text{s.t.} \quad & w^*(\lambda) \in \arg\min_{w \in \mathbb{R}^d} \mathcal{F}(w, \lambda) \end{aligned} \tag{6.11}$$

where $\mathcal{C}$ denotes a compact convex subset, $\mathcal{U} : \mathbb{R}^d \times \mathcal{C} \to \mathbb{R}$ and $\mathcal{F} : \mathbb{R}^d \times \mathcal{C} \to \mathbb{R}$ are continuously differentiable functions. Problem (6.11) is a rewriting of (6.10) with $\mathcal{C} = \Delta_s$, $\mathcal{U}(w, \lambda) = \mathcal{U}(w)$, and $\mathcal{F}(w, \lambda) = \sum_{a \in \mathcal{A}} \lambda_a F_a(w)$. Applied to problem (6.11), our algorithm BADR-GD reads:

$$\begin{aligned} w_{t+1} &\leftarrow w_t - \tau \nabla_w \mathcal{F}(w_t, \lambda_t) \\ v_{t+1} &\leftarrow v_t - \rho\big(\nabla_{w,w} \mathcal{F}(w_t, \lambda_t) v_t + \nabla_w \mathcal{U}(w_t, \lambda_t)\big) \\ \lambda_{t+1} &\leftarrow \text{Proj}_{\mathcal{C}}\Big(\lambda_t - \gamma\Big(\nabla^2_{w,\lambda} \mathcal{F}(w_t, \lambda_t)^\top v_t + \nabla_\lambda \mathcal{U}(w_t, \lambda_t)\Big)\Big) \end{aligned} \tag{6.12}$$

As such, it can be seen as a variant of the SOBA method (Dagréou et al., 2022), with its parameterization adapted to the structure of (6.10). Indeed, in addition to the novel constraint on the upper variable $\lambda$, the main distinction from existing work is the absence of global smoothness of the lower-level objective $\mathcal{F}$ with respect to $\lambda$. In fact, in the specific setting of Problem (6.10), $\mathcal{F}$ admits the form $\mathcal{F}(w, \lambda) = \sum_{s \in \mathcal{A}} \lambda_s F_s(w)$. This leads to mixed derivatives of the form:

$$\nabla^2_{w,\lambda}\mathcal{F}(w,\lambda) = [\nabla F_1(w) \;\; \nabla F_2(w) \;\; \cdots \;\; \nabla F_S(w)]. \tag{6.13}$$

Hence, uniform smoothness cannot hold, since the gradients of group losses $\nabla F_s$ are unbounded over $\mathbb{R}^d$ due to the strong convexity of the functions $F_s$. This in turn invalidates the previous convergence analysis of SOBA (Dagréou et al., 2022), in several places including the Lipschitz continuity of the inner solution $w^* : \lambda \mapsto \arg\min_w \mathcal{F}(w,\lambda)$, the smoothness of the implicit function $\phi : \lambda \mapsto \mathcal{U}(w^*(\lambda),\lambda)$, and the control of the error in the implicit-gradient estimate. In contrast, our approach relies on weaker assumptions on the lower-level objective $\mathcal{F}$ that remain reasonable in practice and that we clarify below.

**Assumption 6.4.** The upper objective $\mathcal{U}$ satisfies the following conditions:

1. $\mathcal{U}$ is Lipschitz continuous on $\mathbb{R}^d \times \mathcal{C}$ with constant $L_{\mathcal{U},0} > 0$, i.e. for all $w_1, w_2 \in \mathbb{R}^d$ and $\lambda_1, \lambda_2 \in \mathcal{C}$:
$$|\mathcal{U}(w_1,\lambda_1) - \mathcal{U}(w_2,\lambda_2)| \leqslant L_{\mathcal{U},0}\|(w_1,\lambda_1) - (w_2,\lambda_2)\| \tag{6.14}$$
2. $\nabla\mathcal{U}$ is Lipschitz continuous on $\mathbb{R}^d \times \mathcal{C}$ with Lipschitz constant $L_{\mathcal{U},1} > 0$, i.e. for all $w_1, w_2 \in \mathbb{R}^d$ and $\lambda_1, \lambda_2 \in \mathcal{C}$:
$$\|\nabla_w\mathcal{U}(w_1,\lambda_1) - \nabla_w\mathcal{U}(w_2,\lambda_2)\| \leqslant L_{\mathcal{U},1}\|(w_1,\lambda_1) - (w_2,\lambda_2)\| \tag{6.15}$$

Assumption 6.4 is standard in the bilevel optimization literature (Arbel & Mairal, 2021; Dagréou et al., 2022; Ghadimi & Wang, 2018) and is essential to ensure global smoothness of the implicit function $\phi : \lambda \mapsto \mathcal{U}(w^*(\lambda),\lambda)$. We complement it with the following assumption on the lower-level objective.

**Assumption 6.5.** The lower objective $\mathcal{F}$ is twice continuously differentiable and satisfies the following conditions:

1. $\mathcal{F}$ is strongly convex with respect to $w$ on $\mathbb{R}^d \times \mathcal{C}$ with strong convexity constant $\mu_{\mathcal{F}} > 0$. That is, for all $w_1, w_2 \in \mathbb{R}^d$ and $\lambda \in \mathcal{C}$:
$$\mathcal{F}(w_2,\lambda) \geqslant \mathcal{F}(w_1,\lambda) + \langle \nabla_w\mathcal{F}(w_1,\lambda), w_2 - w_1\rangle + \frac{\mu_{\mathcal{F}}}{2}\|w_2 - w_1\|^2 \tag{6.16}$$
2. $\nabla_w\mathcal{F}$ is Lipschitz with respect to $w$ on $\mathbb{R}^d$ with Lipschitz constant $L^{w,w}_{\mathcal{F},1} \geqslant 0$, uniformly over $\mathcal{C}$. That is, for all $w_1, w_2 \in \mathbb{R}^d$ and $\lambda \in \mathcal{C}$:
$$\|\nabla_w\mathcal{F}(w_1,\lambda) - \nabla_w\mathcal{F}(w_2,\lambda)\| \leqslant L^{w,w}_{\mathcal{F},1}\|w_1 - w_2\|. \tag{6.17}$$
3. $\nabla^2_{w,w}\mathcal{F}$ is Lipschitz continuous on $\mathbb{R}^d \times \mathcal{C}$ with Lipschitz constant $L^{w,w}_{\mathcal{F},2} \geqslant 0$. That is, for all $w_1, w_2 \in \mathbb{R}^d$ and $\lambda_1, \lambda_2 \in \mathcal{C}$:
$$\left\|\nabla^2_{w,w}\mathcal{F}(w_1,\lambda_1) - \nabla^2_{w,w}\mathcal{F}(w_2,\lambda_2)\right\| \leqslant L^{w,w}_{\mathcal{F},2}\|(w_1,\lambda_1) - (w_2,\lambda_2)\|. \tag{6.18}$$

4. $\nabla^2_{w,\lambda}\mathcal{F}$ is uniformly upper bounded by a constant $M_{\mathcal{C}} > 0$ over the solution set $\{(w^*(\lambda_1), \lambda_2), (\lambda_1, \lambda_2) \in \mathcal{C}^2\}$, i.e. for all $\lambda_1, \lambda_2 \in \mathcal{C}$,

$$\left\|\nabla^2_{w,\lambda}\mathcal{F}(w^*(\lambda_1), \lambda_2)\right\| \leqslant M_{\mathcal{C}}. \tag{6.19}$$

5. $\nabla^2_{w,\lambda}\mathcal{F}$ is Lipschitz with constant $L^{w,\lambda}_{\mathcal{F},2} > 0$. That is, for all $w_1, w_2 \in \mathbb{R}^d$ and $\lambda_1, \lambda_2 \in \mathcal{C}$:

$$\left\|\nabla^2_{w,\lambda}\mathcal{F}(w_1, \lambda_1) - \nabla^2_{w,\lambda}\mathcal{F}(w_2, \lambda_2)\right\| \leqslant L^{w,\lambda}_{\mathcal{F},2}\|(w_1, \lambda_1) - (w_2, \lambda_2)\|. \tag{6.20}$$

The first three conditions of Assumption 6.5 are also standard assumptions for ensuring the well-posedness of the solution mapping $\lambda \mapsto w^*(\lambda)$ and the smoothness of the implicit function $\phi$. A notable distinction from existing work (Arbel & Mairal, 2021; Dagréou et al., 2022; Lu, 2023) is the absence of global smoothness of the lower objective with respect to $\lambda$. Instead, we relax this assumption and only require an upper bound for $\nabla^2_{w,\lambda}\mathcal{F}$ over the solution set $\{(w^*(\lambda), \lambda), \lambda \in \mathcal{C}\}$. Finally, we further require this mixed derivative to be Lipschitz continuous, which would directly follow from a smoothness assumption on the gradients $\nabla F_s$ in (6.10). We note that these assumptions are typically satisfied in learning problems with strongly convex loss functions and standard regularization schemes, such as ridge regression or logistic regression, provided the data are bounded[17].

Our first convergence result is stated in the following theorem. The convergence metric we consider is a positive combination of the squared norm of the generalized gradient

$$\bar{D}_{\lambda,t} \triangleq \frac{1}{\gamma}(\lambda_t - \mathrm{Proj}_{\mathcal{C}}(\lambda_t - \gamma\nabla\phi(\lambda_t))) \tag{6.21}$$

and the squared distance of the lower variable to optimality $\|w_t - w^*(\lambda_t)\|^2$. The generalized gradient vanishes if and only if $\lambda_t$ is a first-order stationary point of the implicit objective $\phi$ over $\mathcal{C}$, that is, no feasible direction decreases $\phi$ to first order. The second term measures the distance of $w_t$ to the exact lower-level solution $w^*(\lambda_t)$. Since the left-hand side of the theorem is an average, at least one iterate satisfies both criteria up to $\mathcal{O}(1/T)$. For (6.10), this means that no small change of the group weights yields a fairer model to first order, and that $w_t$ is close to the Pareto-efficient model associated with $\lambda_t$. Finally, we denote by $\epsilon_{w,0} \triangleq \|w_0 - w^*(\lambda_0)\|$ and $\epsilon_{v,0} \triangleq \|v_0 - v^*(\lambda_0, w_0)\|$ the initialization errors on the lower and dual variables, respectively.

[17] Our extension to the stochastic setting in Theorem 6.8 relaxes this requirement by allowing data with only bounded second-order moments.

**Theorem 6.6.** *Consider running the iteration (6.12) for $T$ steps with step sizes $\tau, \rho, \gamma$ satisfying $\tau = \rho = 1/L_{\mathcal{F},1}^{w,w}$ and $\gamma = \min\left(\kappa_{\mathcal{F},w}^{-1}, \bar{\gamma}\right)$ where $\kappa_{\mathcal{F},w} \triangleq L_{\mathcal{F},1}^{w,w}/\mu_{\mathcal{F}}$ and $\bar{\gamma}$ is a constant made explicit in (6.100). Then, under Assumption 6.4 and Assumption 6.5, the sequence $(w_t, \lambda_t)_{t\geqslant 0}$ generated by (6.12) satisfies for all $T \geqslant 1$,*

$$\frac{1}{T}\sum_{t=0}^{T-1}\Big[\underbrace{\|\bar{D}_{\lambda,t}\|^2}_{\substack{\text{upper-level}\\ \text{stationarity}}} + \bar{\alpha}\underbrace{\|w^*(\lambda_t) - w_t\|^2}_{\substack{\text{lower-level}\\ \text{optimality}}}\Big] \leqslant \frac{8}{\gamma T}\Big(\Phi(\lambda_0) - \inf_{\lambda\in\mathcal{C}}\Phi(\lambda)\Big) + \frac{4\kappa_{\mathcal{F},w}}{T}\Big(\bar{\alpha}\epsilon_{w,0}^2 + \bar{\beta}\epsilon_{v,0}^2\Big) \tag{6.22}$$

*where*

$$\bar{\alpha} = 12\Big(L_{\mathcal{U},1}^{w,\lambda} + L_{\mathcal{F},2}^{w,\lambda}\Big(2\kappa_{\mathcal{F},w}\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}} + \epsilon_{v,0}\Big)\Big)^2 + 2568\kappa_{\mathcal{F},w}^2 M_{\mathcal{C}}^2\Big(L_{\mathcal{U},0}\frac{L_{\mathcal{F},2}^{w,w}}{\mu_{\mathcal{F}}^2} + \frac{L_{\mathcal{U},1}}{\mu_{\mathcal{F}}}\Big)^2 \tag{6.23}$$
$$\bar{\beta} = 6M_{\mathcal{C}}^2.$$

We defer the proof of this theorem to Section 6.B of the appendix.

**Remark.** Our convergence rate is $\mathcal{O}(1/T)$ in squared gradient norm, which matches the standard rate for nonconvex smooth optimization (Beck, 2017) as well as the rate achieved for bilevel algorithms (Dagréou et al., 2022). Furthermore, our convergence metric incorporates optimality at the lower-level, an improvement over the analysis derived in Dagréou et al. (2022).

**Remark.** An important component of our analysis is the control of the dual variable $v_t$ along the iterations. Specifically, we bound $\|v_t\|$ uniformly over time, using the strong convexity of the lower-level problem. This in turn enables us to control the error $\left\|\nabla\Phi(\lambda_t) - \left(\nabla_{w,\lambda}^2\mathcal{F}(w_t,\lambda_t)^\top v_t + \nabla_\lambda\mathcal{U}(w_t,\lambda_t)\right)\right\|$ on the gradient of the implicit function.

We complement our analysis with convergence guarantees for BADR-SGD, our stochastic variant of BADR-GD defined in Algorithm 6.2. BADR-SGD allows for stochastic estimates of the oracles involved in (6.10). Unlike in the deterministic setting, we cannot uniformly control the dual variable $v_t$ along the iterations as in the deterministic analysis; instead, we can only bound its second moment (see Lemma 6.21). Although uniform boundedness of the dual variable plays a key role in controlling the stochastic error of the implicit gradient estimate $\tilde{\nabla}_{w,\lambda}^2\mathcal{F}(w_t,\lambda_t)^\top v_t + \tilde{\nabla}_\lambda\mathcal{U}(w_t,\lambda_t)$, we show that this issue can be effectively handled by introducing an additional clipping mechanism. Following our formalism, we consider now iterations of the form:

$$
\begin{aligned}
w_{t+1} &\leftarrow w_t - \tau \tilde{\nabla}_w \mathcal{F}(w_t, \lambda_t) \\
v_{t+1} &\leftarrow v_t - \rho\Big(\tilde{\nabla}_{w,w} \mathcal{F}(w_t, \lambda_t) v_t + \tilde{\nabla}_w \mathcal{U}(w_t, \lambda_t)\Big) \\
\lambda_{t+1} &\leftarrow \mathrm{Proj}_{\mathcal{C}}\Big(\lambda_t - \gamma\, \mathbf{Clip}\Big(\tilde{\nabla}^2_{w,\lambda} \mathcal{F}(w_t, \lambda_t)^\top v_t + \tilde{\nabla}_\lambda \mathcal{U}(w_t, \lambda_t)\Big)\Big)
\end{aligned}
\tag{6.24}
$$

where $\tilde{\nabla}$ denotes a stochastic estimate of the corresponding gradient, and $\mathbf{Clip} : g \mapsto \min\big(1, C_\gamma / \|g\|\big) g$ denotes a clipping operator and $C_\gamma$ is an additional hyper-parameter of our method.

We assume that stochastic estimates are revealed independently at each iteration, and we denote by $\mathfrak{S}_t$ the filtration generated by the random variables up to iteration $t$. Let us further introduce the intermediate $\sigma$-algebras $\mathfrak{S}_{t,w} = \sigma(\mathfrak{S}_t, w_{t+1})$ and $\mathfrak{S}_{t,v} = \sigma\big(\mathfrak{S}_{t,w}, v_{t+1}\big)$. We complement the previous assumptions with the following standard conditions on our stochastic estimates.

**Assumption 6.7.** The stochastic gradients $\tilde{\nabla}_w \mathcal{F}$, $\tilde{\nabla}_{w,w} \mathcal{F}$, $\tilde{\nabla}^2_{w,\lambda} \mathcal{F}$, $\tilde{\nabla}_w \mathcal{U}$ and $\tilde{\nabla}_\lambda \mathcal{U}$ are independent unbiased estimates of the corresponding true gradients, with bounded variance. That is, for all $w \in \mathbb{R}^d$ and $\lambda \in \mathcal{C}$:

$$
\begin{aligned}
&\mathbb{E}\Big[\tilde{\nabla}_w \mathcal{F}(w_t, \lambda_t) \mid \mathfrak{S}_t\Big] = \nabla_w \mathcal{F}(w, \lambda) \qquad \mathbb{E}\Big[\Big\|\tilde{\nabla}_w \mathcal{F}(w, \lambda) - \nabla_w \mathcal{F}(w, \lambda)\Big\|^2 \mid \mathfrak{S}_t\Big] \leqslant \mathcal{V}_{\mathcal{F},1}, \\
&\mathbb{E}\Big[\tilde{\nabla}^2_{w,w} \mathcal{F}(w, \lambda) \mid \mathfrak{S}_{t,w}\Big] = \nabla^2_{w,w} \mathcal{F}(w, \lambda) \mathbb{E}\Big[\Big\|\tilde{\nabla}^2_{w,w} \mathcal{F}(w, \lambda) - \nabla^2_{w,w} \mathcal{F}(w, \lambda)\Big\|^2 \mid \mathfrak{S}_{t,w}\Big] \leqslant \mathcal{V}^{w,w}_{\mathcal{F},2}, \\
&\mathbb{E}\Big[\tilde{\nabla}^2_{w,\lambda} \mathcal{F}(w, \lambda) \mid \mathfrak{S}_{t,v}\Big] = \nabla^2_{w,\lambda} \mathcal{F}(w, \lambda) \;\; \mathbb{E}\Big[\Big\|\tilde{\nabla}^2_{w,\lambda} \mathcal{F}(w, \lambda) - \nabla^2_{w,\lambda} \mathcal{F}(w, \lambda)\Big\|^2 \mid \mathfrak{S}_{t,v}\Big] \leqslant \mathcal{V}^{w,\lambda}_{\mathcal{F},2} \\
&\mathbb{E}\Big[\tilde{\nabla}_w \mathcal{U}(w, \lambda) \mid \mathfrak{S}_{t,w}\Big] = \nabla_w \mathcal{U}(w, \lambda) \qquad \mathbb{E}\Big[\Big\|\tilde{\nabla}_w \mathcal{U}(w, \lambda) - \nabla_w \mathcal{U}(w, \lambda)\Big\|^2 \mid \mathfrak{S}_{t,w}\Big] \leqslant \mathcal{V}^{w}_{\mathcal{U},1}, \\
&\mathbb{E}\Big[\tilde{\nabla}_\lambda \mathcal{U}(w, \lambda) \mid \mathfrak{S}_{t,v}\Big] = \nabla_\lambda \mathcal{U}(w, \lambda) \qquad \mathbb{E}\Big[\Big\|\tilde{\nabla}_\lambda \mathcal{U}(w, \lambda) - \nabla_\lambda \mathcal{U}(w, \lambda)\Big\|^2 \mid \mathfrak{S}_{t,v}\Big] \leqslant \mathcal{V}^{\lambda}_{\mathcal{U},1}.
\end{aligned}
\tag{6.25}
$$

where $\mathcal{V}_{\mathcal{F},1}, \mathcal{V}^{w,w}_{\mathcal{F},2}, \mathcal{V}^{w,\lambda}_{\mathcal{F},2}, \mathcal{V}^{w}_{\mathcal{U},1}, \mathcal{V}^{\lambda}_{\mathcal{U},1} > 0$ are positive constants.

The following theorem states our convergence result for BADR-SGD. Besides the initial error $\epsilon_{w,0}$ and $\epsilon_{v,0}$, we also introduce $\epsilon^2_{w,v,0} \triangleq \|w_0 - w^*(\lambda_0)\|^2 \|v_0\|^2$ whose motivation is stated in the following remarks.

**Theorem 6.8.** *Consider running the iteration (6.24) for $T$ steps with step sizes $\tau, \rho, \gamma$ satisfying $\tau = 1/L^{w,w}_{\mathcal{F},1}$, $\rho = \min\Big(1/L^{w,w}_{\mathcal{F},1}, \frac{\mu_{\mathcal{F}}}{4\mathcal{V}^{w,w}_{\mathcal{F},2}}\Big)$, and $\gamma = \min\Big(\bar{\bar{\gamma}}, L_\phi^{-1}\Big)$ where $L_\phi$ denotes the smoothness constant of the implicit loss $\phi$ and $\bar{\bar{\gamma}}$ is a constant made explicit in (6.138). Then, under Assumption 6.4, Assumption 6.5, and Assumption 6.7, the sequence $(w_t, \lambda_t)_{t \geqslant 0}$ generated by (6.24) satisfies for all $T \geqslant 1$,*

$$\frac{1}{T}\sum_{t=0}^{T-1}\left[\mathbb{E}\left[\left\|\bar{D}_{\lambda,t}\right\|\right]+\tilde{\alpha}^{1/2}\mathbb{E}\left[\left\|w_t-w_t^*\right\|\right]\right]\leqslant \mathcal{S}\left[\frac{8\Delta\phi}{\gamma T}+\frac{32\kappa_{\mathcal{F}}}{T}\left(\tilde{\alpha}\epsilon_{w,0}^2+\tilde{\beta}\epsilon_{v,0}^2+\tilde{\delta}\epsilon_{w,v,0}^2\right)+\mathbf{M}^\top\mathcal{V}+rC_\gamma^2\right] \quad (6.26)$$

*where* $\mathcal{S}[t]\triangleq t/C_\gamma+\sqrt{t}$, $\Delta\phi\triangleq\phi(\lambda_0)-\inf_{\lambda\in\mathcal{C}}\phi(\lambda)$, *and* $\tilde{\alpha},\tilde{\beta},\tilde{\delta}$ *are problem-dependent constants. The vector* $\mathbf{M}\in\mathbb{R}^5$ *is a problem-dependent vector, the scalar* $r$ *is defined as* $r\triangleq 18L_\phi^{-2}M_{\mathcal{C}}^2\left(L_{\mathcal{F},2}^{w,w}\right)^2\mathcal{C}_{v,*}\mu_{\mathcal{F}}^{-4}$, *and* $\mathcal{V}\triangleq\left[\mathcal{V}_{\mathcal{U},1}^{\lambda},\mathcal{V}_{\mathcal{U},1}^{w},\mathcal{V}_{\mathcal{F},1},\mathcal{V}_{\mathcal{F},2}^{w,\lambda},\mathcal{V}_{\mathcal{F},2}^{w,w}\right]^\top$ *collects the variances specified in Assumption 6.7.*

The proof of this theorem is given in Section 6.B.2.

**Remark.** In Theorem 6.8, the convergence metric jointly quantifies optimality at the implicit and lower levels. Regarding optimality at the implicit level, the metric used is the expected norm of the stochastic generalized gradient $\mathbb{E}\left[\left\|\bar{D}_{\lambda,t}\right\|\right]$, which slightly differs from the standard metric in stochastic nonconvex proximal optimization (Ghadimi et al., 2016), which would instead involve $\mathbb{E}\left[\left\|D_{\lambda,t}\right\|^2\right]$, the expected squared norm of the generalized stochastic gradient. Besides having the expectation outside of the norm—which we argue better characterizes stationarity because randomness concerns only the path up to $\lambda_t$, not the stochastic gradient at time $t$—we note that our results present convergence in gradient norm instead of squared gradient norm, which is a common feature in analyses involving clipping (Koloskova et al., 2023).

**Remark.** The right-hand side of our convergence rate involves three components. The first two components respectively bound the speed at which we forget the initial conditions and the variance of our stochastic estimates. We note that the matrix $\mathbf{M}$ depends on the problem constants introduced in Assumption 6.4, and does not vanish as $\gamma$ goes to zero. Hence, to achieve an $\epsilon$-stationary point, one may resort to mini-batching to reduce the noise vector $\mathcal{V}$ to an appropriate norm. In particular, setting both the minibatch size $m$ and the number of iterations $T$ to scale as $\epsilon^{-2}$ yields an overall complexity of order $\mathcal{O}\left(\epsilon^{-4}\right)$, which matches the known lower bound with respect to $\epsilon$ for stochastic bilevel problems (Chen & Zhang, 2025). We do not, however, investigate optimality with respect to the condition number $\kappa$, as our framework addresses a broader class of problems, and establishing corresponding lower bounds in this regime lies beyond the scope of the present work.

**Remark.** We prove Theorem 6.8 based on the derivation of a novel Lyapunov function $\psi_t$ defined as

$$\psi_t\triangleq\mathbb{E}\Big[\underbrace{\phi(\lambda_t)}_{\text{implicit objective}}+\alpha\underbrace{\|w_t-w^*(\lambda_t)\|^2}_{\text{lower-level error}}+\beta\underbrace{\|v_t-v^*(\lambda_t)\|^2}_{\text{dual error}}+\delta\underbrace{\|w_t-w^*(\lambda_t)\|^2\|v_t\|^2}_{\text{coupling}}\Big] \quad (6.27)$$

where $\alpha, \beta, \delta > 0$ are carefully tuned parameters. Besides the implicit objective, $\psi_t$ tracks the errors on the lower and dual variables, which together control the error on the implicit gradient. The decrease of $\psi_t$ is enabled by the clipping of the upper-level update, which specifically allows us to control the drift of the fourth term $\mathbb{E}\left[\delta\|w_t - w^*(\lambda_t)\|^2\|v_t\|^2\right]$ of our Lyapunov function. This term is responsible for the presence of the additional starting error $\epsilon^2_{w,v,0}$ in our convergence rate.

## 6.4 Open-source software

We provide `badr`, an open-source Python package that implements the adaptive reweighting methodology presented in Section 6.2. `badr` follows the `scikit-learn` interface, allowing users to train and make predictions with a model through simple function calls directly from `pandas` DataFrames. In the following, we describe the package's interface and give simple examples of a `badr` workflow. We refer to the online documentation for more details and advanced usage of the package.

### 6.4.1 Interface

We show the class structure of the `badr` package in Figure 6.2 below.

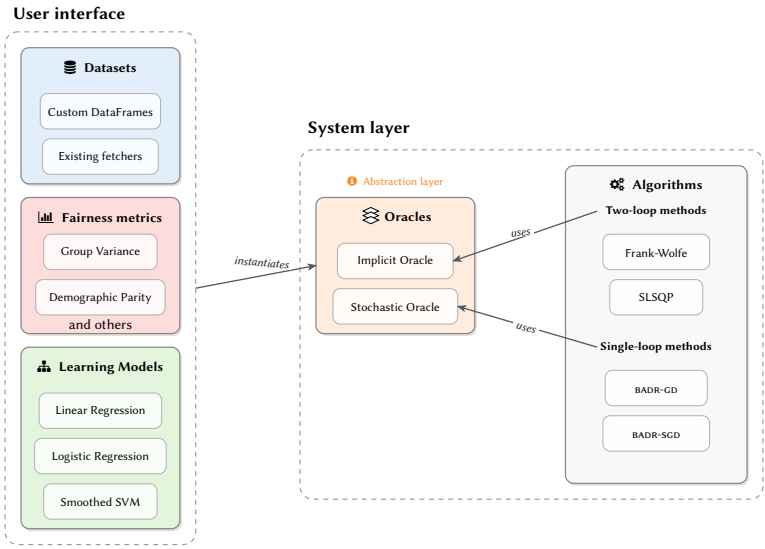



Figure 6.2: **Overview of the `badr` package modules.**

The package separates the machine learning task (consisting of a dataset, a fairness metric, and a learning model) from the algorithms used to solve the adaptive reweighting problem as stated in Section 6.3.

### 6.4.2 Data

Users of `badr` are not expected to prepare custom data loaders or write custom training loops. They can instead use the following three lines with a `pandas` DataFrame to create a Dataset object, perform a train/test split, normalize the feature matrix, and partition the data according to the specified sensitive attributes.

```
import pandas as pd
from badr.datasets import load_dataframe
df = pd.read_csv(...) # load your dataset
dset = load_dataframe(df = df, target_col = "y", sensitive_cols
= ["sex"])
```

### 6.4.3 Basic usage

We start by importing the `badr` package and defining a problem, i.e., a dataset, a learning model, and a fairness metric.

```
import badr as bdr

# == Problem Definition ==#
dset = bdr.datasets.fetch_adult()
model = bdr.models.LogisticRegression()
metric = bdr.metrics.IndividualFairness()
```

We can follow the `scikit-learn` API using the `badr` estimators. As an example, we show below the fitting and scoring of the `scikit-learn` LogisticRegression estimator.

```
#== NO FAIRNESS ENFORCED ==#
model.fit(dset.X_train, dset.y_train, dset.groups)
test_score = model.score(dset.X_test, dset.y_test)
test_fairness = metric.fun(model.coef_, dset)
```

Building a Pareto-fair counterpart to the `scikit-learn` model can be done with one additional line, as shown below.

```
#== BADR ==#
badr = bdr.Badr(dset, model, metric)
badr.run()
badr_test_score = badr.model.score(dset.X_test, dset.y_test)
badr_fairness = metric.fun(badr.coef_, dset)
```

To fit the Pareto-fair model, we first instantiate a `badr` object with the dataset, the learning model, and the fairness metric and call `.run()`. Depending on the size of the dataset, either an implicit or a stochastic oracle is instantiated, which serves as an abstraction layer between the fairness problem and the proposed algorithms.

The `.run()` method performs two tasks: (1) it executes a suitable optimization algorithm to compute an adaptive reweighting and (2) it fits the learning model with the computed per-group weights.

### 6.4.4 Algorithms

The `badr` package mainly includes implementations of Algorithm 6.1 and Algorithm 6.2, corresponding to the large-scale algorithms BADR-GD and BADR-SGD for solving the bilevel problem (6.10). We also allow the use of two-loop algorithms to solve small- and medium-scale instances of (6.10).

Two-loop methods for bilevel optimization work as follows: at each iteration, the lower-level problem is (approximately) solved using an iterative method (inner loop). The resulting solution is used to compute an implicit gradient, after which the outer variable is updated iteratively (outer loop). This is a standard approach to solving bilevel optimization problems (Domke, 2012; Ghadimi & Wang, 2018), but it can be computationally expensive and less scalable than single-loop algorithms because the lower-level problem must be solved multiple times and solving the adjoint equation to compute the gradient of the solution mapping typically requires a matrix inversion.

Default choices for the optimization method In the remainder of this section, we present the optimization methods available in the package for two-loop algorithms. We outline the default parameters and stopping criteria for each method. We identify the default two-loop method and justify this choice. We discuss how the wall-clock time of each method scales with an increasing number of groups. Finally, we compare the scalability of the default two-loop method with BADR-SGD.

**Implementation.** We implement the two-loop algorithms by (i) solving the lower-level problem with the `scikit-learn` models used in our experiments (ridge regression, $\ell_2^2$-regularized logistic regression, and an $\ell_2^2$-regularized smoothed SVM) and (ii) updating the outer variable using an implicit gradient. For logistic regression and smoothed SVM, we obtain the lower-level gradient and Hessian with `jax` (Bradbury et al., 2018) and solve the resulting linear system with NUMPY for ridge regression, we solve it directly with CVXPY (Agrawal et al., 2018; Diamond & Boyd, 2016) and the CLARABEL solver (Goulart & Chen, 2024) (default settings).

**Available optimization methods.** As outlined in Section 6.4.4, (6.10) can either be solved using Algorithm 6.1 and Algorithm 6.2 or recast as the minimization of a smooth nonconvex function on the unit probability simplex with available function and gradient evaluations. We consider three optimization methods to solve this problem: SLSQP (Kraft, 1994), the trust-region method (Conn et al., 2000), and the

Frank-Wolfe method (Braun et al., 2022). Both SLSQP and the trust-region method are implemented in SciPy, with SLSQP calling a C backend. For Frank-Wolfe, we adapt the reference implementation from the book repository (Braun et al., 2022) to support JAX arrays and just-in-time compilation, and we use the oblivious step size $\frac{2}{t+2}$ instead of backtracking (Pedregosa et al., 2020), since repeated implicit-gradient evaluations dominated any gains from line-search step sizes. Other methods, such as projected gradient descent, mirror descent, an augmented Lagrangian method, and the pairwise Frank-Wolfe variant, were discarded in preliminary experiments. The first three were much slower than the methods above unless their step sizes were carefully tuned for each instance, while the pairwise variant was slower than standard Frank-Wolfe. Moreover, in fairness applications, the simplex has at most a few dozen vertices, so that the cost of each iteration is dominated by the lower-level solves. In such settings, exploiting the simplex geometry through a mirror map or a linear minimization oracle is unlikely to yield significant speedups.

**Stopping criteria.** Each of the three optimization methods has its own stopping criterion, in addition to a maximum iteration limit of 500. SLSQP stops when the absolute difference between consecutive function values falls below $10^{-5}$. The trust-region method stops when the norm of the Lagrangian gradient falls below $10^{-5}$. The Frank-Wolfe method stops when the Frank-Wolfe gap (Braun et al., 2022, Definition 1.11) (which upper-bounds the current iterate's suboptimality) falls below $10^{-6}$.

**Default method.** In this section, we justify our selection of SLSQP as the default optimization method for the two-loop approach. We conducted a systematic experiment on the 10 largest subsamples of the 2014 ACS Employment dataset (the 10 most populated U.S. states). Each subsample was partitioned into two to six sensitive groups, yielding 50 (state, number of groups) instances. We used the Disparate Mistreatment (Zafar et al., 2017) fairness metric. For each instance, we ran all three solvers and recorded the best fairness value $\mathcal{U}_{\text{best}}$ attained by any solver. We then defined the target precision as $\varepsilon = \mathcal{U}_{\text{best}} + \epsilon$ and measured the wall-clock time to reach this target across all five partitionings (two to six groups). We provide performance profiles for three precision values, $\epsilon = 10^{-3}$, $10^{-5}$, and $10^{-7}$, in Figure 6.3.

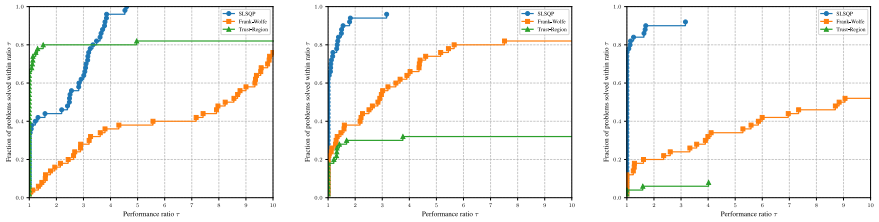



Figure 6.3: Performance profiles comparing SLSQP, Frank-Wolfe, and the trust-region method for achieving $\mathcal{U}_{\texttt{best}} + \epsilon$ across multiple tolerance levels. The vertical axis represents the fraction of instances solved, while the horizontal axis shows the factor $\tau$ of wall-clock time relative to the fastest algorithm. Higher curves indicate superior algorithm efficiency. Rightward-extending curves identify instances where algorithms demonstrate slower convergence relative to the optimal method.

Our main findings are summarized below:

- **SLSQP consistently solves the most instances.** SLSQP outperforms both Frank-Wolfe and the trust-region method in terms of fraction of problems solved across all precision values. SLSQP achieves a 100% success rate for low precision ($\epsilon = 10^{-3}$) and approximately 94% for moderate and high precision ($\epsilon \in \{10^{-5}, 10^{-7}\}$). In contrast, although the trust-region method solves around 80% of instances at low precision, its performance degrades considerably when asking for higher precisions.
- **SLSQP remains within a factor of three of the fastest method for moderate and high precision.** For $\epsilon \in \{10^{-5}, 10^{-7}\}$, approximately 90% of problems are solved by SLSQP with $\tau < 2$, indicating it rarely requires more than twice the computational time of the best solver. When slower, it typically remains within $\tau \approx 3$.

Scalability We evaluate how the runtime of the proposed methods evolves as problem size increases in two dimensions: the number of sensitive groups and the sample size.

**Groups.** We use a dataset derived from the 2018 ACSTravelTime data (Ding et al., 2021) with up to 36 groups, obtained by crossing age ($< 20$ vs. $\geqslant 20$), race (nine categories), and gender (male and female). We predict whether an individual's commute exceeds 20 minutes (the 2018 U.S. median) with an $\ell_2^2$-regularized logistic regression model and the Disparate Mistreatment metric. For each `badr` instantiation, we record the wall-clock time to first reach a fairness value below $10^{-5}$ on subsamples from five U.S. states (Alaska, Alabama, Colorado, Massachusetts, and Wyoming). Results are shown in Figure 6.4.

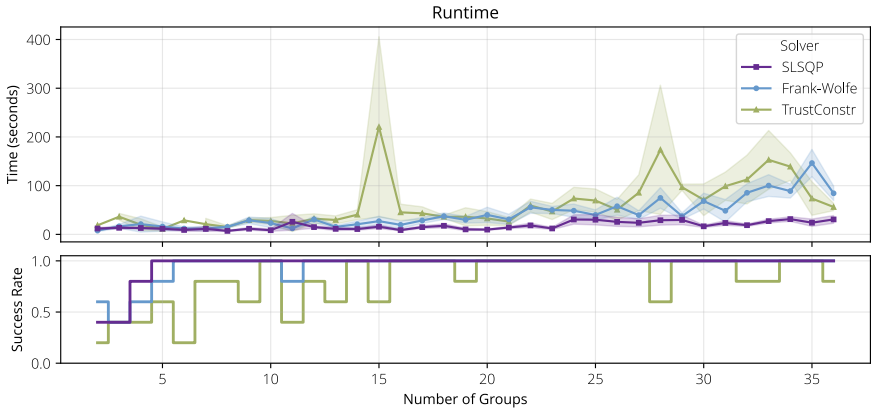



Figure 6.4: (Top): Mean wall-clock time (solid) with the standard error of the mean (shaded) computed over successful runs across five U.S. states for SLSQP, Frank-Wolfe, and the trust-region method to first reach a fairness value below $10^{-5}$, as a function of the number of sensitive groups. (Bottom): Success rate of each method over the same five states as a function of the number of sensitive groups.

For fewer than 10 groups, SLSQP and Frank-Wolfe have similar mean runtime. The trust-region method is comparable on successful runs but fails much more often for 2 to 5 groups. Beyond 10 groups, SLSQP remains nearly constant in mean runtime, with a noticeably narrower standard-error band, whereas Frank-Wolfe and the trust-region method slow down. Finally, SLSQP's success rate is never below the others except at 2 groups (2 successful runs for SLSQP vs. 3 for Frank-Wolfe). These results motivate our choice of SLSQP as the default optimizer for the two-loop bilevel approach.

**Sample size.** We compare the two-loop method (with SLSQP) and our stochastic algorithm BADR-SGD by scaling the sample size, as reported in Figure 6.5.

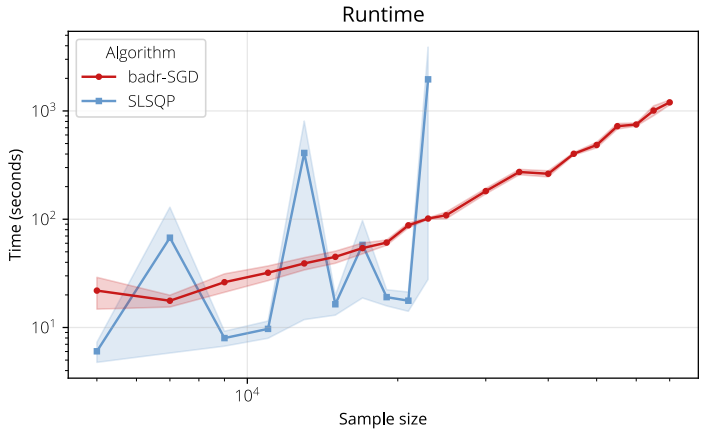



Figure 6.5: Runtime scaling of BADR-SGD and SLSQP with dataset size (mean $\pm$ standard deviation over five seeds).

Both methods train a fair logistic regression model with $\ell_2^2$-regularization parameter $10^{-1}$ on gender-partitioned subsamples of the California ACSEmployment dataset. For each sample size, we draw 5 subsamples (different seeds) and report mean training time and standard deviation. Experiments were run on a laptop with 32 GB of RAM. SLSQP ran out of memory at $25,000$ samples, indicating that it can be competitive while it fits in memory but becomes impractical as sample size increases.

## 6.5 NUMERICAL EXPERIMENTS

In this section, we provide a detailed description of the experiments conducted to analyze the effectiveness of the adaptive reweighting methodology. In Section 6.5.1, we describe the datasets used and the problem formulations considered. In Section 6.5.2.1 and Section 6.5.2.2, we examine whether BADR provides fairer solutions than other baselines. In Section 6.5.2.3, we study how BADR compares with other baselines in terms of test error.

### 6.5.1 Datasets, tasks, and methods

Datasets We consider eleven datasets, summarized in Table 6.1 and described below in detail.

Table 6.1: Summary table of all real-world datasets considered. The sample size ($n$) and number of features ($d$) are reported after preprocessing.

| Task | Dataset | Target | Sensitive Attribute ($S$) | $n$ | $d$ |
|---|---|---|---|---|---|
| Classification | ACSEmployment | Employment | Gender (2), Race (9) | 3157599 | 17 |
| | ACSIncome | Income $\geqslant$ 50K | Gender (2), Race (9) | 1664500 | 11 |
| | ACSTravelTime | Commute $\geqslant$ 20min | Gender (2), Age (2), Race (9) | 1376075 | 16 |
| | Adult | Income $\geqslant$ 50K | Gender (2), Race (5) | 32561 | 109 |
| | COMPAS | Recidivism in 2Y | Gender (2), Race (6) | 6172 | 9 |
| | German Credit | Credit (Good/Bad) | Gender (2) | 1000 | 65 |
| | Arrhythmia | Arrhythmia (Yes/No) | Gender (2) | 418 | 272 |
| Regression | Law School (LSAC) | GPA | Gender (2), Race (5) | 22387 | 10 |
| | Parkinsons Telemonitoring | UPDRS Score | Gender (2) | 5875 | 21 |
| | Communities and Crime | Violent Crimes | Gender (2), Race (4) | 1993 | 97 |
| | Student Performance | Final Grade | Gender (2) | 395 | 54 |

The three ACS datasets (ACSEmployment, ACSIncome, and ACSTravelTime) are preprocessed following the guidelines in Ding et al. (2021), while the remaining datasets use the preprocessing strategy described in Do et al. (2022). All datasets are standardized to have zero mean and unit variance. Additionally, when using a linear model, we append an *intercept* column of ones to the feature matrix. For the three largest datasets (ACSEmployment, ACSIncome, ACSTravelTime), we also create smaller, subsampled versions by partitioning the original data by state (50 states + Puerto Rico) and year (2014–2018), yielding $51 \times 5 = 255$ smaller datasets with sizes ranging between 3,000 and 150,000 samples. For large-scale experiments, we further partition the ACS datasets by the four statistical regions defined by the U.S. Census Bureau (Northeast, Midwest, West, South), resulting in datasets containing between 300,000 and 600,000 samples. In total, we obtain 785 datasets. Each dataset can be partitioned into sensitive groups, with the number of groups ranging from 2 to 36, depending on the attributes considered. Moreover, each partitioned dataset can be analyzed using one of many unfairness metrics (we consider 7 different metrics). This gives a large benchmark suite and allows us to evaluate both small-scale and large-scale scenarios.

Learning models We consider three common models for supervised learning, whose losses we describe below.

- **$\ell_2^2$-regularized logistic regression:**

$$\ell(f(w, x_i), y_i) = \log(1 + \exp(-y_i \langle w, x_i \rangle)) + \frac{\rho}{2}\|w\|^2 \tag{6.28}$$

  This model is used for binary classification tasks.
- **$\ell_2^2$-regularized smoothed SVM:**

$$\ell(f(w, x_i), y_i) = \max\{0, 1 - y_i \langle w, x_i \rangle\}^2 + \frac{\rho}{2}\|w\|^2 \tag{6.29}$$

This model is also used for binary classification tasks.
- **Ridge regression:**

$$\ell(f(w, x_i), y_i) = (y_i - \langle w, x_i \rangle)^2 + \frac{\rho}{2}\|w\|^2 \tag{6.30}$$

  This model is used for regression tasks.

Note that $\rho$ is a hyperparameter that controls the regularization strength.

Methods BADR outputs a set of group weights $(\lambda_a^*)_{a \in \mathcal{A}} \in \Delta_{\mathcal{S}}$ and the associated fitted model $w^*(\lambda^*)$. For each unfairness metric $\mathcal{U}$, we compare five models obtained through the following strategies:

a) **Adaptive reweighting strategy:** we train the model using `badr`, our adaptive reweighting framework;
b) **Uniform sampling strategy:** we train the model without group reweighting; the group-weight vector is the barycenter $\frac{1}{|\mathcal{S}|}\mathbf{1}$ of the simplex $\Delta_{\mathcal{S}}$;
c) **Balanced sampling strategy:** we train the model using group weights that rebalance the dataset, assigning each group a weight inversely proportional to its sample size and normalizing them to unit sum;
d) **One-group fitting strategy:** we separately train one model per sensitive group and keep the fairest model, that is, the model trained on group $\arg\min_{a \in \mathcal{A}} \mathcal{U}(w^*(e_a))$;
e) **Minimax fairness:** we train the model following the minimax fairness approach (Abernethy et al., 2020; Diana et al., 2021; Hashimoto et al., 2018; Martinez et al., 2020), by solving the optimization problem defined in (6.8).

Recall that these five strategies provide a Pareto-efficient model in terms of the group losses, and that we are mostly interested in whether the model trained with BADR is fairer than other Pareto-efficient models.

Hyperparameters The hyperparameters of Algorithm 6.1 are $\tau$ and $\gamma$ (note that we can set $\rho = \tau$, see Theorem 6.6). We select both hyperparameters using a coarse-to-fine grid search.

Metric used We evaluate the models trained by each method on two aspects: (i) prediction performance, measured by classification error for binary classification and root-mean-square error (RMSE) for regression, and (ii) the unfairness value of the trained model.

Hardware All experiments were run on a laptop with an Intel i7 processor and 32 GB of RAM, using `badr` and its dependencies. The experiments were conducted on the datasets described in Section 6.5.1.

### 6.5.2 Experimental results

In this subsection, we provide numerical examples showing that solutions produced by BADR achieve lower unfairness than other Pareto-efficient solutions. We first analyze the specific cases of two and three groups in Section 6.5.2.1. For these smaller group configurations, we visualize the implicit fairness function and plot the solutions returned by BADR alongside baseline methods. We then examine the general case in Section 6.5.2.2, plotting fairness value histograms on train and test sets for each unfairness metric across all five strategies (BADR, minimax fairness, one-group fitting, balanced sampling, and uniform sampling). In Section 6.5.2.3, we turn our attention to the effect of the reweighting strategy on predictive performance.

Experiment 1: Comparison with baselines on two and three groups In this experiment, we visualize the value of the unfairness metric $\mathcal{U}$ with varying group weights for two and three groups using the $\ell_2^2$-regularized logistic regression model. For two groups, we consider three different metrics and three different datasets, and we compare BADR with baselines under a gender-based partition. Figure 6.6 illustrates our findings.

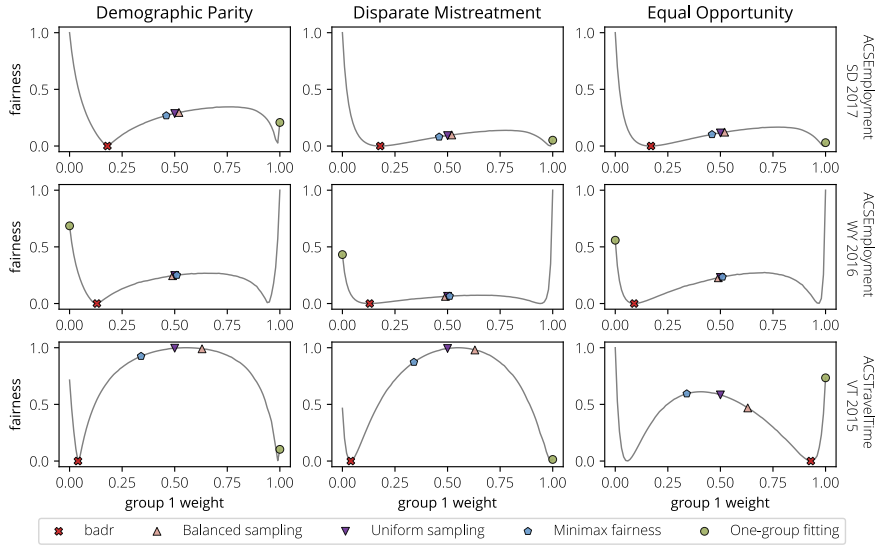



Figure 6.6: Visualization of the implicit fairness function for two groups for different metrics (columns) and different datasets (rows). Grey lines map the normalized fairness measure, representing the relative distance from the minimum achievable fairness value (y-axis) for varying group weights (x-axis). Markers indicate solutions found by each baseline.

The implicit fairness function represents the value of the unfairness metric $\mathcal{U}$ over the Pareto front of group losses.

**Optimization over the Pareto front is meaningful.** We first observe that, for each unfairness metric and dataset, the implicit fairness function varies, motivating the approach of optimizing an unfairness metric over the Pareto front.

**Mismatch between the sampling provided by the baselines and the unfairness metric.** We observe that the implicit function contains multiple local minima, and that the optimal sampling obtained using `badr` is nontrivial and never attained by any of the baseline methods. Moreover, simple and widely used heuristics such as balanced sampling (McMahan et al., 2017) do not necessarily improve fairness over standard uniform sampling.

These observations are consistent on datasets partitioned into three groups, for which we propose a visualization in Figure 6.7.

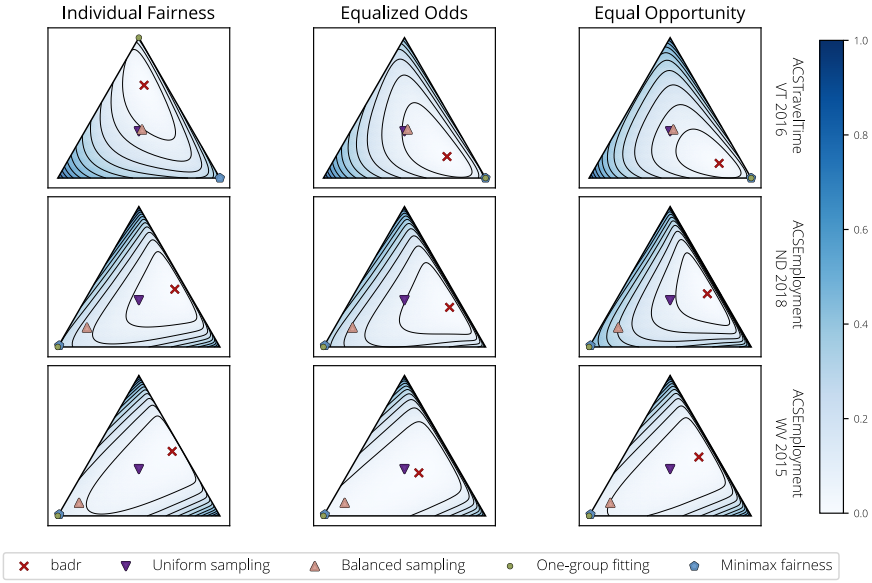



Figure 6.7: Normalized unfairness metric across three datasets (rows) and three metrics (columns) as a function of the group weights. Markers indicate solutions found by `badr` and baselines. The three-dimensional unit probability simplex is plotted as an equilateral triangle through barycentric coordinate transformation.

Experiment 2: Fairness attained on datasets partitioned in an arbitrary number of groups For an arbitrary number of groups, it is no longer possible to visualize the value of the unfairness metric $\mathcal{U}$ as a function of the group weights. In this experiment, we evaluate the models trained by each strategy (BADR, uniform sampling, balanced sampling, one-group fitting, and minimax fairness) in terms of the value of the unfairness metric $\mathcal{U}$ on both the train and test sets. Our evaluation uses the ACSEmployment dataset for classification tasks. We systematically examine all subsamples of the dataset (organized by state and year) and evaluate unfairness metrics across every possible sensitive group partitioning. For each unfairness metric relevant to classification tasks, we record the fairness values achieved by `badr` and baseline methods. We cluster results by unfairness metric type. Figure 6.8 shows CDF plots for each of the five strategies across both training and test sets.

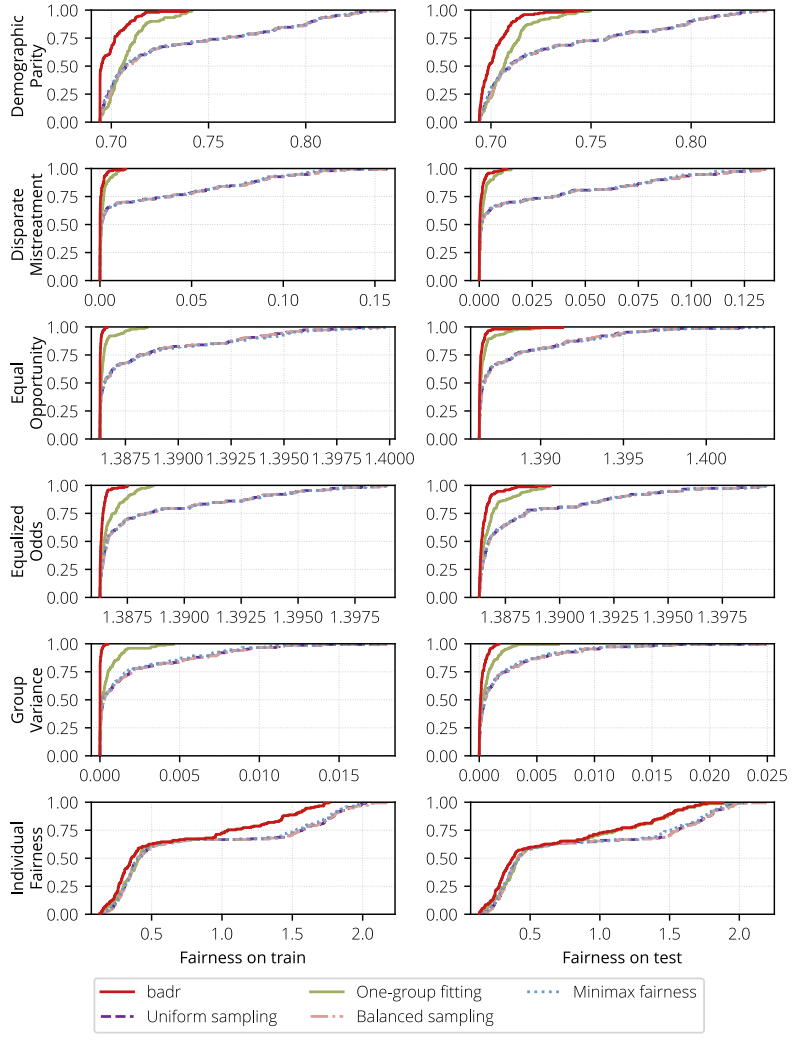



Figure 6.8: Empirical cumulative distribution functions (ECDFs) of six unfairness metrics (demographic parity, disparate mistreatment, equal opportunity, equalized odds, group variance, and individual fairness), computed using five approaches: BADR, uniform sampling, balanced sampling, one-group fitting, and minimax fairness. Curves are shown separately for the training data (left) and the test data (right). The horizontal axes show metric values, and the vertical axes show the empirical cumulative probabilities (fraction of runs with unfairness no larger than a given value). For each metric, curves that rise more quickly and lie closer to the top-left indicate better fairness performance.

Our main findings are summarized below.

- **badr consistently yields improved fairness across all metrics.** BADR and minimax fairness both yield fairness value distributions tightly clustered on the fair end (left), outperforming uniform sampling and one-group fitting across all metrics.
- **badr is the most robust in terms of fairness on the test set.** The weights obtained by BADR still yield the best fairness on the test set. This contrasts with minimax fairness, which exhibits a wider right tail on the test set for several metrics: equal opportunity, demographic parity, and group variance.
- **One-group fitting shows large variation and poor fairness performance.** In both training and test sets, one-group fitting produces wide ranges of fairness values. Its results differ markedly from the tight clusters formed by BADR, minimax fairness, and uniform sampling.
- **Uniform sampling achieves moderate fairness consistency but performs worse than badr and minimax fairness.** Uniform sampling matches BADR and minimax fairness for individual fairness. For other unfairness metrics, it underperforms these two methods. Overall, it controls fairness better than one-group fitting but less effectively than BADR and minimax fairness.
- **Fairness performance stability differs between training and testing.** All methods show degradation in fairness when moving from training to test data. BADR is less affected by this degradation than the other approaches. One-group fitting is systematically worse. Minimax fairness degrades notably for the demographic parity metric. Uniform sampling, while robust, performs less consistently than BADR and minimax fairness.

Experiment 3: Accuracy–fairness trade-off In this experiment, we evaluate the accuracy–fairness trade-off of the five Pareto-efficient strategies in Section 6.5.1.3 on classification and regression tasks. For classification, we use 51 subsamples each from ACSEmployment and ACSTravelTime (102 datasets total), partition each dataset by gender, and apply a 70/30 train–test split; we use logistic regression ($\ell_2^2$-regularization weight $\rho = 10^{-2}$) and individual fairness, and report mean test accuracy and mean test unfairness with standard deviations over the 102 datasets. For regression, we use Law School, Parkinson's Telemonitoring, Communities and Crime, and Student Performance, with the same gender partitioning and 70/30 train–test split. We use ridge regression ($\ell_2^2$-regularization weight $\rho = 10^{-1}$), RMSE, and demographic parity, and report mean test RMSE and mean test unfairness with standard deviations averaged over four datasets, each with ten random train/test splits. The results of the experiment are shown in Table 6.2.

Table 6.2: Test accuracy and individual unfairness (classification; 102 runs) and test RMSE and demographic parity unfairness (regression; 40 runs), reported as mean ± standard deviation.

| | Classification (102 runs) | | Regression (40 runs) | |
|---|---|---|---|---|
| Method | Test accuracy (↑) | Test unfairness (↓) | Test RMSE (↓) | Test unfairness (↓) |
| BADR | 0.7628 ± 0.1231 | **0.2947 ± 0.1031** | 2.5554 ± 4.1021 | **0.8666 ± 0.1188** |
| Uniform sampling | 0.7637 ± 0.1219 | 0.3432 ± 0.1089 | 2.5085 ± 4.0420 | 1.0974 ± 0.3672 |
| Balanced sampling | 0.7634 ± 0.1221 | 0.3470 ± 0.1102 | 2.5168 ± 4.0550 | 1.0935 ± 0.3596 |
| Minimax fairness | 0.7633 ± 0.1223 | 0.3372 ± 0.1077 | 2.5314 ± 4.0855 | 1.0728 ± 0.3453 |
| One-group fitting | 0.7585 ± 0.1282 | 0.3180 ± 0.1152 | 2.6800 ± 4.3114 | 1.0239 ± 0.2766 |

**Better fairness without a loss of accuracy.** All five baselines exhibit comparable mean predictive performance: mean test accuracy is very similar across methods for classification, and mean test RMSE is similarly close for regression, with differences that are minor compared with the reported standard deviations. In contrast, fairness varies considerably. BADR consistently yields the lowest test unfairness in both settings (0.2947 vs. 0.3180-0.3470 for classification, and 0.8666 vs. 1.02-1.10 for regression), indicating improved fairness at comparable predictive performance.

## 6.6 CONCLUSION

In this chapter, we introduced BADR, a bilevel framework for fair learning that selects, among groupwise Pareto-efficient models, the model that optimizes a user-specified fairness metric. We achieve this by treating the group scalarization weights as variables learned through a bilevel formulation: the lower level enforces Pareto efficiency of the group losses, while the upper level optimizes the chosen unfairness criterion. This perspective makes it possible to target a specific fairness metric while retaining efficiency, whereas the effect of fixed reweighting schemes or robust minimax objectives on a given metric is difficult to control. From an optimization standpoint, we developed scalable single-loop algorithms for this bilevel problem and established their convergence under mild regularity condi-

tions, despite the lack of global smoothness induced by the coupling between the weights and the group losses. We complemented these theoretical results with `badr`, an open-source toolbox implementing our approach for several fairness metrics and standard learning models, and we demonstrated on real datasets that BADR improves the targeted metric over Pareto-efficient baselines while preserving Pareto-efficient group performance.

This framework opens several research directions at the intersection of multi-objective optimization, bilevel optimization, and statistical fairness. The most important one concerns the statistical understanding of metric-driven selection over the Pareto front: the out-of-sample behaviour of the selected fairness metric and the extent to which improvements observed on the empirical Pareto front transfer to test performance.

## 6.A FAIRNESS METRICS

Table 6.3: Task type handled by each fairness metric considered.

| Reference | Fairness metric | Classification | Regression |
|---|---|---|---|
| (Berk et al., 2017) | Individual Fairness | ✓ | ✓ |
| (Calders et al., 2009) | Demographic Parity | ✓ | ✓ |
| (Zafar et al., 2017) | Disparate Mistreatment | ✓ | ✗ |
| (Hardt et al., 2016) | Equal Opportunity | ✓ | ✗ |
| (Hardt et al., 2016) | Equalized Odds | ✓ | ✗ |
| (Gretton et al., 2005) | Hilbert-Schmidt Independence Criterion | ✗ | ✓ |
| · | Group Variance | ✓ | ✓ |

BADR relies on differentiable unfairness metrics to enable its bilevel adaptive procedure.

In this section, we introduce and describe the unfairness metrics implemented and used in our numerical experiments. These metrics, summarized in Table 6.3, are defined as follows:

- *group variance* corresponds to the variance of the group losses, that is

$$\mathcal{U}_{\mathrm{GV}}(w) = \frac{1}{S}\sum_{a=1}^{S}\left(F_a(w) - \frac{1}{S}\sum_{b=1}^{S}F_b(w)\right)^2. \tag{6.31}$$

- *individual fairness* penalizes models for how differently they treat $x_i$ and $x_j$ when these two samples come from different sensitive groups while having similar outcomes. We generalize the definition for two groups proposed in Berk et al. (2017) to multiple groups as:

$$\mathrm{IF}(w) \triangleq \frac{1}{\sum_{1\leqslant a<b\leqslant g} n_a n_b} \sum_{1\leqslant a<b\leqslant g} \sum_{i:a_i=a} \sum_{j:a_j=b} e^{-|y_i-y_j|}\big(f(w,x_i)-f(w,x_j)\big)^2 \quad (6.32)$$

- *demographic parity* (Calders et al., 2009) requires the predictions of a machine learning model to be independent of sensitive attributes. For classification tasks, it can be defined as $\mathcal{U}_{\mathrm{DP}}(w) = \max_{a\in\mathcal{A}} |p_a(w) - \overline{p}(w)|$, where $p_a(w) = n_a^{-1} \sum_{i:a_i=a} \mathbf{1}\{f(w,x_i) \geqslant 0\}$, and $\overline{p}(w) = g^{-1} \sum_{a\in\mathcal{A}} p_a(w)$. For regression tasks, we instead use the group mean predictions, $p_a(w) = n_a^{-1} \sum_{i:a_i=a} f(w,x_i)$. To alleviate the nonsmoothness of this metric, we implement the following differentiable approximation:

$$\mathcal{U}_{\mathrm{DP}}^{\rho}(w) = \frac{1}{\rho} \log\left( \sum_{a\in\mathcal{A}} \Big( \exp\Big(\rho\Big(p_a^{(\rho)}(w) - \overline{p}^{(\rho)}(w)\Big)\Big) + \exp\Big(-\rho\Big(p_a^{(\rho)}(w) - \overline{p}^{(\rho)}(w)\Big)\Big)\Big)\right) \quad (6.33)$$

  where, for classification tasks, $p_a^{(\rho)}(w) = n_a^{-1} \sum_{i:a_i=a} \sigma(\rho f(w,x_i))$, while for regression tasks, $p_a^{(\rho)}(w) = n_a^{-1} \sum_{i:a_i=a} f(w,x_i)$, and $\overline{p}^{(\rho)}(w) = g^{-1} \sum_{a\in\mathcal{A}} p_a^{(\rho)}(w)$. Here, $\sigma$ denotes the sigmoid function and $\rho > 0$ is a smoothing parameter.
- *disparate mistreatment* is a convex and differentiable variant of demographic parity introduced in Zafar et al. (2017). It is defined as $\mathcal{U}_{\mathrm{DM}}(w) = \big(\frac{1}{n}\sum_{i=1}^{n}(a_i - \bar{a})\big(f(w,x_i) - \bar{f}(w)\big)\big)^2$.
- *equalized odds* and *equal opportunities* are specific to classification tasks. Equalized odds, defined as $\mathcal{U}_{\mathrm{EOd}}(w) = \frac{1}{2}(\max_{a\in\mathcal{A}} \mathrm{TPR}_a(w) - \min_{a\in\mathcal{A}} \mathrm{TPR}_a(w) + \max_{a\in\mathcal{A}} \mathrm{FPR}_a(w) - \min_{a\in\mathcal{A}} \mathrm{FPR}_a(w))$, requires both the true positive and false positive rates to be equal across groups (Hardt et al., 2016). Alternatively, equal opportunity can be written as $\mathcal{U}_{\mathrm{EOp}}(w) = \max_{a\in\mathcal{A}} \mathrm{TPR}_a(w) - \min_{a\in\mathcal{A}} \mathrm{TPR}_a(w)$ and only requires true positive rates to be equal across groups. To handle their non-differentiability, we define smoothed counterparts to true positive and false positive rates as

$$\mathrm{TPR}_a^{\rho}(w) \triangleq \frac{1}{n_{a,1}} \sum_{i:a_i=a,y_i=1} \sigma(\rho f(w,x_i)), \quad \mathrm{FPR}_a^{\rho}(w) \triangleq \frac{1}{n_{a,0}} \sum_{i:a_i=a,y_i=0} \sigma(\rho f(w,x_i)), \quad (6.34)$$

  where $\rho > 0$ is a smoothing parameter, $n_{a,1} = \#\{i : a_i = a, y_i = 1\}$ and $n_{a,0} = \#\{i : a_i = a, y_i = 0\}$. This leads to the differentiable approximations:

$$\begin{aligned}\mathcal{U}^{\rho}_{\mathrm{EOp}}(w) &\triangleq \frac{1}{\rho}\log\left(\sum_{a\in\mathcal{A}}\exp\left(\rho\,\mathrm{TPR}^{\rho}_{a}(w)\right)\right)+\frac{1}{\rho}\log\left(\sum_{a\in\mathcal{A}}\exp\left(-\rho\,\mathrm{TPR}^{\rho}_{a}(w)\right)\right),\\ \mathcal{U}^{\rho}_{\mathrm{EOd}}(w) &\triangleq \frac{1}{2}\left(\mathcal{U}^{\rho}_{\mathrm{EOp}}(w)+\frac{1}{\rho}\log\left(\sum_{a\in\mathcal{A}}\exp\left(\rho\,\mathrm{FPR}^{\rho}_{a}(w)\right)\right)+\frac{1}{\rho}\log\left(\sum_{a\in\mathcal{A}}\exp\left(-\rho\,\mathrm{FPR}^{\rho}_{a}(w)\right)\right)\right).\end{aligned} \tag{6.35}$$

- *Hilbert-Schmidt Independence Criterion (HSIC)* (Gretton et al., 2005) measures statistical dependence between two random variables via RKHS embeddings. In our experiments, we use a linear kernel and it reduces to the squared Hilbert-Schmidt norm of the empirical cross-covariance:

$$\mathcal{U}_{\mathrm{HSIC}}(w) \triangleq \left\|\frac{1}{n-1}\sum_{i=1}^{n}(s_i-\bar{s})\big(f(w,x_i)-\bar{f}(w)\big)\right\|_2^2, \tag{6.36}$$

where $\bar{s}\triangleq\frac{1}{n}\sum_{i=1}^{n}s_i$, $\bar{f}(w)\triangleq\frac{1}{n}\sum_{i=1}^{n}f(w,x_i)$, $n\triangleq\sum_{a\in\mathcal{A}}n_a$.

## 6.B Convergence Analysis

This section is dedicated to the proof of our main convergence results, namely Theorem 6.6 and Theorem 6.8.

### 6.B.1 The deterministic case

In this section, we present the proof of Theorem 6.6.

We first derive some regularity properties of the implicit objective $\phi:\lambda\mapsto\mathcal{U}(w^*(\lambda),\lambda)$ in Section 6.B.1.1.

Sections 6.B.1.2 and 6.B.1.3 control the lower and dual updates, respectively.

These properties enable us to control the error in the approximate gradient of $\phi$ in Section 6.B.1.4.

Finally, we combine all these results to prove Theorem 6.6 in Section 6.B.1.5 using a properly parameterized Lyapunov function.

Throughout, $\lambda_t, w_t, v_t$ denote the iterates generated by (6.12) at iteration $t\geqslant 0$.

Optimality on the lower level is measured through the squared distance $\|w_t-w^*(\lambda_t)\|^2$ to the optimal solution $w^*(\lambda_t)\triangleq\arg\min_w\mathcal{F}(w,\lambda_t)$, while optimality on the upper level is measured through the squared norm $\|\bar{D}_{\lambda,t}\|^2$ of the generalized gradient

$$\bar{D}_{\lambda,t}\triangleq\frac{1}{\gamma}(\lambda_t-\mathrm{Proj}_{\mathcal{C}}(\lambda_t-\gamma\nabla\phi(\lambda_t))). \tag{6.37}$$

We further introduce the approximate gradient $\mathrm{D}^{\mathrm{in}}_{\lambda,t}$ and the approximate generalized gradient $\mathrm{D}_{\lambda,t}$ as

$$\begin{aligned}\mathrm{D}^{\mathrm{in}}_{\lambda,t} &\triangleq \nabla_\lambda \mathcal{U}(w_t,\lambda_t) + \nabla^2_{w,\lambda}\mathcal{F}(w_t,\lambda_t)^\top v_t,\\ \mathrm{D}_{\lambda,t} &\triangleq \frac{1}{\gamma}\Big(\lambda_t - \mathrm{Proj}_{\mathcal{C}}\Big(\lambda_t - \gamma \mathrm{D}^{\mathrm{in}}_{\lambda,t}\Big)\Big).\end{aligned} \tag{6.38}$$

Note that the upper level update in (6.12) can be rewritten as $\lambda_{t+1} = \lambda_t - \gamma \mathrm{D}_{\lambda,t} = \mathrm{Proj}_{\mathcal{C}}\Big(\lambda_t - \gamma \mathrm{D}^{\mathrm{in}}_{\lambda,t}\Big)$. Finally, we also introduce the dual solutions $v^*(\lambda_t)$ and $v^*(\lambda_t, w_t)$ as

$$\begin{aligned}v^*(\lambda_t, w_t) &\triangleq -\nabla^2_{w,w}\mathcal{F}(w_t,\lambda_t)^{-1}\nabla_w \mathcal{U}(w_t,\lambda_t),\\ v^*(\lambda_t) &\triangleq v^*(\lambda_t, w^*(\lambda_t)) = -\nabla^2_{w,w}\mathcal{F}(w^*(\lambda_t),\lambda_t)^{-1}\nabla_w \mathcal{U}(w^*(\lambda_t),\lambda_t).\end{aligned} \tag{6.39}$$

Smoothness of the implicit map $\lambda \mapsto \mathcal{U}(w^*(\lambda),\lambda)$

In this paragraph, we derive regularity results on the solution map $\lambda \mapsto w^*(\lambda)$ as well as on the implicit objective $\lambda \mapsto \mathcal{U}(w^*(\lambda),\lambda)$.

We begin with a Lipschitz-continuity result for $w^*(\cdot)$.

**Lemma 6.9.** (Lipschitz-continuity of the solution map) *Under Assumption 6.5, the solution map $\lambda \mapsto w^*(\lambda)$ is $\frac{M_{\mathcal{C}}}{\mu_{\mathcal{F}}}$-Lipschitz. That is, for any $\lambda_1, \lambda_2 \in \mathbb{R}^{d_\lambda}$, $\|w^*(\lambda_1) - w^*(\lambda_2)\| \leqslant \frac{M_{\mathcal{C}}}{\mu_{\mathcal{F}}}\|\lambda_1 - \lambda_2\|$.*

*Proof.* By strong convexity of $\mathcal{F}$ with respect to $w$, we have for any $\lambda_1, \lambda_2 \in \mathbb{R}^{d_\lambda}$,

$$\begin{aligned}\frac{\mu_{\mathcal{F}}}{2}\|w^*(\lambda_2) - w^*(\lambda_1)\|^2 &\leqslant \mathcal{F}(w^*(\lambda_2),\lambda_2) - \mathcal{F}(w^*(\lambda_1),\lambda_2)\\ &\quad -\langle \nabla_w \mathcal{F}(w^*(\lambda_1),\lambda_2), w^*(\lambda_2) - w^*(\lambda_1)\rangle,\\ \frac{\mu_{\mathcal{F}}}{2}\|w^*(\lambda_2) - w^*(\lambda_1)\|^2 &\leqslant \mathcal{F}(w^*(\lambda_1),\lambda_2) - \mathcal{F}(w^*(\lambda_2),\lambda_2)\\ &\quad -\langle \nabla_w \mathcal{F}(w^*(\lambda_2),\lambda_2), w^*(\lambda_1) - w^*(\lambda_2)\rangle.\end{aligned} \tag{6.40}$$

Hence, summing the two inequalities, we obtain

$$\begin{aligned}\mu_{\mathcal{F}}\|w^*(\lambda_2) - w^*(\lambda_1)\|^2 &\leqslant \langle \nabla_w \mathcal{F}(w^*(\lambda_2),\lambda_2) - \nabla_w \mathcal{F}(w^*(\lambda_1),\lambda_2), w^*(\lambda_2) - w^*(\lambda_1)\rangle.\\ &= \langle \nabla_w \mathcal{F}(w^*(\lambda_1),\lambda_2), w^*(\lambda_1) - w^*(\lambda_2)\rangle.\end{aligned} \tag{6.41}$$

The equality follows from the optimality condition $\nabla_w \mathcal{F}(w^*(\lambda_2),\lambda_2) = 0$. Moreover, using Assumption 6.5 (item 4), we have

$$\|\nabla_w \mathcal{F}(w^*(\lambda_1),\lambda_2)\| = \|\nabla_w \mathcal{F}(w^*(\lambda_1),\lambda_2) - \nabla_w \mathcal{F}(w^*(\lambda_1),\lambda_1)\| \leqslant M_{\mathcal{C}}\|\lambda_1 - \lambda_2\|. \tag{6.42}$$

By the Cauchy–Schwarz inequality, we obtain

$$\mu_{\mathcal{F}}\|w^*(\lambda_2) - w^*(\lambda_1)\|^2 \leqslant M_{\mathcal{C}}\|\lambda_1 - \lambda_2\| \cdot \|w^*(\lambda_2) - w^*(\lambda_1)\|. \tag{6.43}$$

Hence, $\|w^*(\lambda_2) - w^*(\lambda_1)\| \leqslant \frac{M_{\mathcal{C}}}{\mu_{\mathcal{F}}}\|\lambda_1 - \lambda_2\|$. □

Let us now show the smoothness of the implicit objective $\phi$.

**Lemma 6.10.** (Smoothness of the implicit objective) *The implicit objective* $\phi : \lambda \mapsto \mathcal{U}(w^*(\lambda), \lambda)$ *is* $L_\phi$*-smooth with*

$$L_\phi \triangleq \left( L_{\mathcal{U},1}\left(\frac{M_{\mathcal{C}}}{\mu_{\mathcal{F}}} + 1\right) + L_{\mathcal{U},0}\left(\frac{L_{\mathcal{F},2}^{w,w} M_{\mathcal{C}}}{\mu_{\mathcal{F}}^2} + \frac{L_{\mathcal{F},2}^{w,\lambda}}{\mu_{\mathcal{F}}}\right)\right)\left(\frac{M_{\mathcal{C}}}{\mu_{\mathcal{F}}} + 1\right). \tag{6.44}$$

*Proof.* Let us first establish the smoothness of $w^*(\cdot)$. Using the strong convexity of $\mathcal{F}$ with respect to $w$, and in turn the invertibility of the Hessian $\nabla^2_{w,w}\mathcal{F}$, the implicit function theorem guarantees differentiability of $w^*(\cdot)$ for all $\lambda \in \mathcal{C}$,

$$\operatorname{Jac} w^*(\lambda) = -\nabla^2_{w,w}\mathcal{F}(w^*(\lambda), \lambda)^{-1}\nabla^2_{w,\lambda}\mathcal{F}(w^*(\lambda), \lambda). \tag{6.45}$$

Hence, for any $\lambda_1, \lambda_2 \in \mathcal{C}$, we have:

$$\begin{aligned}\|\operatorname{Jac} w^*(\lambda_1) - \operatorname{Jac} w^*(\lambda_2)\| \leqslant{}& \left\|\nabla^2_{w,w}\mathcal{F}(w^*(\lambda_1), \lambda_1)^{-1} - \nabla^2_{w,w}\mathcal{F}(w^*(\lambda_2), \lambda_2)^{-1}\right\| \cdot \left\|\nabla^2_{w,\lambda}\mathcal{F}(w^*(\lambda_1), \lambda_1)\right\| \\ &+ \left\|\nabla^2_{w,w}\mathcal{F}(w^*(\lambda_2), \lambda_2)^{-1}\right\| \cdot \left\|\nabla^2_{w,\lambda}\mathcal{F}(w^*(\lambda_1), \lambda_1) - \nabla^2_{w,\lambda}\mathcal{F}(w^*(\lambda_2), \lambda_2)\right\|.\end{aligned} \tag{6.46}$$

By Assumption 6.5 (item 4), we have $\left\|\nabla^2_{w,\lambda}\mathcal{F}(w^*(\lambda_1), \lambda_1)\right\| \leqslant M_{\mathcal{C}}$. Moreover, using Assumption 6.5 (item 1), we have the upper bound $\left\|\nabla^2_{w,w}\mathcal{F}(w^*(\lambda_2), \lambda_2)^{-1}\right\| \leqslant \frac{1}{\mu_{\mathcal{F}}}$. Thus, combined with the matrix identity $A^{-1} - B^{-1} = A^{-1}(B - A)B^{-1}$, we obtain

$$\begin{aligned}\left\|\nabla^2_{w,w}\mathcal{F}(w^*(\lambda_1), \lambda_1)^{-1} - \nabla^2_{w,w}\mathcal{F}(w^*(\lambda_2), \lambda_2)^{-1}\right\| &\leqslant \left\|\nabla^2_{w,w}\mathcal{F}(w^*(\lambda_1), \lambda_1)^{-1}\right\| \cdot \left\|\nabla^2_{w,w}\mathcal{F}(w^*(\lambda_2), \lambda_2)^{-1}\right\| \\ &\quad \cdot \|\nabla^2_{w,w}\mathcal{F}(w^*(\lambda_2), \lambda_2) - \nabla^2_{w,w}\mathcal{F}(w^*(\lambda_1), \lambda_1)\| \\ &\leqslant \mu_{\mathcal{F}}^{-2}\|\nabla^2_{w,w}\mathcal{F}(w^*(\lambda_2), \lambda_2) - \nabla^2_{w,w}\mathcal{F}(w^*(\lambda_1), \lambda_1)\| \\ &\leqslant L_{\mathcal{F},2}^{w,w}\mu_{\mathcal{F}}^{-2}(\|w^*(\lambda_1) - w^*(\lambda_2)\| + \|\lambda_1 - \lambda_2\|)\end{aligned} \tag{6.47}$$

where the last inequality follows from Assumption 6.5 (item 3). Finally, using Lemma 6.9, we obtain

$$\left\|\nabla^2_{w,w}\mathcal{F}(w^*(\lambda_1), \lambda_1)^{-1} - \nabla^2_{w,w}\mathcal{F}(w^*(\lambda_2), \lambda_2)^{-1}\right\| \cdot \left\|\nabla^2_{w,\lambda}\mathcal{F}(w^*(\lambda_1), \lambda_1)\right\| \leqslant \frac{L_{\mathcal{F},2}^{w,w} M_{\mathcal{C}}}{\mu_{\mathcal{F}}^2}\left(\frac{M_{\mathcal{C}}}{\mu_{\mathcal{F}}} + 1\right)\|\lambda_1 - \lambda_2\|. \tag{6.48}$$

Furthermore, combining Assumption 6.5 (item 1) and Assumption 6.5 (item 5) yields

$$\left\|\nabla^2_{w,w}\mathcal{F}(w^*(\lambda_2),\lambda_2)^{-1}\right\|\left\|\nabla^2_{w,\lambda}\mathcal{F}(w^*(\lambda_1),\lambda_1)-\nabla^2_{w,\lambda}\mathcal{F}(w^*(\lambda_2),\lambda_2)\right\| \leqslant L^{w,\lambda}_{\mathcal{F},2}\mu_{\mathcal{F}}^{-1}(\|w^*(\lambda_1)-w^*(\lambda_2)\| + \|\lambda_1-\lambda_2\|) \tag{6.49}$$

and thus, using again Lemma 6.9, we obtain

$$\left\|\nabla^2_{w,w}\mathcal{F}(w^*(\lambda_2),\lambda_2)^{-1}\right\|\left\|\nabla^2_{w,\lambda}\mathcal{F}(w^*(\lambda_1),\lambda_1)-\nabla^2_{w,\lambda}\mathcal{F}(w^*(\lambda_2),\lambda_2)\right\| \leqslant L^{w,\lambda}_{\mathcal{F},2}\mu_{\mathcal{F}}^{-1}\left(M_{\mathcal{C}}\mu_{\mathcal{F}}^{-1}+1\right)\|\lambda_1-\lambda_2\| \tag{6.50}$$

Summing Equations (6.48) and (6.50), we obtain that $\operatorname{Jac} w^*(\cdot)$ is Lipschitz with constant

$$\|\operatorname{Jac} w^*(\lambda_1)-\operatorname{Jac} w^*(\lambda_2)\| \leqslant \left(\frac{L^{w,w}_{\mathcal{F},2}M_{\mathcal{C}}}{\mu^2_{\mathcal{F}}}+\frac{L^{w,\lambda}_{\mathcal{F},2}}{\mu_{\mathcal{F}}}\right)\left(\frac{M_{\mathcal{C}}}{\mu_{\mathcal{F}}}+1\right)\|\lambda_1-\lambda_2\| \tag{6.51}$$

By Assumption 6.4, $\phi$ is differentiable, and the chain rule gives for any $\lambda \in \mathcal{C}$,

$$\nabla\phi(\lambda) = \nabla_\lambda \mathcal{U}(w^*(\lambda),\lambda) + \operatorname{Jac} w^*(\lambda)^\top \nabla_w \mathcal{U}(w^*(\lambda),\lambda). \tag{6.52}$$

Hence, for any $\lambda_1, \lambda_2 \in \mathcal{C}$, we have

$$\begin{aligned}\|\nabla\phi(\lambda_1)-\nabla\phi(\lambda_2)\| \leqslant{}& \|\nabla_\lambda\mathcal{U}(w^*(\lambda_1),\lambda_1)-\nabla_\lambda\mathcal{U}(w^*(\lambda_2),\lambda_2)\| \\ &+\left\|\operatorname{Jac} w^*(\lambda_1)^\top-\operatorname{Jac} w^*(\lambda_2)^\top\right\|\cdot\|\nabla_w\mathcal{U}(w^*(\lambda_1),\lambda_1)\| \\ &+\left\|\operatorname{Jac} w^*(\lambda_2)^\top\right\|\cdot\|\nabla_w\mathcal{U}(w^*(\lambda_1),\lambda_1)-\nabla_w\mathcal{U}(w^*(\lambda_2),\lambda_2)\|.\end{aligned} \tag{6.53}$$

Combining Assumption 6.4 (item 2) and Lemma 6.9, we have

$$\begin{aligned}\|\nabla_\lambda\mathcal{U}(w^*(\lambda_1),\lambda_1)-\nabla_\lambda\mathcal{U}(w^*(\lambda_2),\lambda_2)\| &\leqslant L_{\mathcal{U},1}(\|\lambda_1-\lambda_2\|+\|w^*(\lambda_1)-w^*(\lambda_2)\|) \\ &\leqslant L_{\mathcal{U},1}\left(\frac{M_{\mathcal{C}}}{\mu_{\mathcal{F}}}+1\right)\|\lambda_1-\lambda_2\|\end{aligned} \tag{6.54}$$

Besides, using (6.51) and Assumption 6.4 (item 1), we have

$$\begin{aligned}&\left\|\operatorname{Jac} w^*(\lambda_1)^\top-\operatorname{Jac} w^*(\lambda_2)^\top\right\|\|\nabla_w\mathcal{U}(w^*(\lambda_1),\lambda_1)\| \\ &\qquad\leqslant L_{\mathcal{U},0}\left(\frac{L^{w,w}_{\mathcal{F},2}M_{\mathcal{C}}}{\mu^2_{\mathcal{F}}}+\frac{L^{w,\lambda}_{\mathcal{F},2}}{\mu_{\mathcal{F}}}\right)\left(\frac{M_{\mathcal{C}}}{\mu_{\mathcal{F}}}+1\right)\|\lambda_1-\lambda_2\|\end{aligned} \tag{6.55}$$

Finally, using Assumption 6.4 (item 2) and Lemma 6.9, we have

$$\left\|\operatorname{Jac} w^*(\lambda_2)^\top\right\|\cdot\|\nabla_w\mathcal{U}(w^*(\lambda_1),\lambda_1)-\nabla_w\mathcal{U}(w^*(\lambda_2),\lambda_2)\| \leqslant \frac{M_{\mathcal{C}}L_{\mathcal{U},1}}{\mu_{\mathcal{F}}}\left(\frac{M_{\mathcal{C}}}{\mu_{\mathcal{F}}}+1\right)\|\lambda_1-\lambda_2\|. \tag{6.56}$$

Summing Equations (6.54), (6.55), and (6.56), we obtain the desired result. □

We finally derive a descent lemma for the implicit objective $\phi$ along the iterates $\lambda_t$.

This lemma will be instrumental in the derivation of our Lyapunov analysis in Section 6.B.1.5.

**Lemma 6.11.** (Descent lemma for the implicit objective) *For any* $t \geqslant 0$*, we have*

$$\phi(\lambda_{t+1}) \leqslant \phi(\lambda_t) - \frac{\gamma}{8}\|\bar{D}_{\lambda,t}\|^2 - \frac{\gamma}{4}\|D_{\lambda,t}\|^2 + \frac{\gamma}{4}\|\bar{D}_{\lambda,t} - D_{\lambda,t}\|^2 + \frac{\gamma}{2}\left\|\nabla\phi(\lambda_t) - D^{\text{in}}_{\lambda,t}\right\|^2 + \frac{\gamma^2 L_\phi}{2}\|D_{\lambda,t}\|^2. \quad (6.57)$$

*Proof.* By $L_\phi$-smoothness of $\phi$, we have for any $t \geqslant 0$,

$$\begin{aligned}\phi(\lambda_{t+1}) &\leqslant \phi(\lambda_t) + \langle \nabla\phi(\lambda_t), \lambda_{t+1} - \lambda_t\rangle + \frac{L_\phi}{2}\|\lambda_{t+1} - \lambda_t\|^2 \\ &= \phi(\lambda_t) - \gamma\langle \nabla\phi(\lambda_t), D_{\lambda,t}\rangle + \frac{\gamma^2 L_\phi}{2}\|D_{\lambda,t}\|^2\end{aligned} \quad (6.58)$$

Since $\lambda_{t+1} = \text{Proj}_{\mathcal{C}}\left(\lambda_t - \gamma D^{\text{in}}_{\lambda,t}\right)$, the characterization of Euclidean projection onto the convex set $\mathcal{C}$ gives $\langle \lambda_t - \gamma D^{\text{in}}_{\lambda,t} - \lambda_{t+1}, \lambda_t - \lambda_{t+1}\rangle \leqslant 0$,

i.e.,

$$\langle D^{\text{in}}_{\lambda,t}, \lambda_{t+1} - \lambda_t\rangle \leqslant -\frac{1}{\gamma}\|\lambda_{t+1} - \lambda_t\|^2 = -\gamma\|D_{\lambda,t}\|^2. \quad (6.59)$$

Hence, using the inequality $\langle a, b\rangle \leqslant \frac{1}{2}\left(\|a\|^2 + \|b\|^2\right)$, we have

$$\begin{aligned}-\gamma\langle \nabla\phi(\lambda_t), D_{\lambda,t}\rangle &= -\gamma\langle D^{\text{in}}_{\lambda,t}, D_{\lambda,t}\rangle + \gamma\langle D^{\text{in}}_{\lambda,t} - \nabla\phi(\lambda_t), D_{\lambda,t}\rangle \\ &\leqslant -\frac{\gamma}{2}\|D_{\lambda,t}\|^2 + \frac{\gamma}{2}\left\|D^{\text{in}}_{\lambda,t} - \nabla\phi(\lambda_t)\right\|^2\end{aligned} \quad (6.60)$$

Plugging this inequality back into the $L_\phi$-smoothness of $\phi$ (6.58), we obtain

$$\begin{aligned}\phi(\lambda_{t+1}) &\leqslant \phi(\lambda_t) - \frac{\gamma}{2}\|D_{\lambda,t}\|^2 + \frac{\gamma}{2}\left\|D^{\text{in}}_{\lambda,t} - \nabla\phi(\lambda_t)\right\|^2 + \frac{\gamma^2 L_\phi}{2}\|D_{\lambda,t}\|^2 \\ &\leqslant \phi(\lambda_t) - \frac{\gamma}{4}\|D_{\lambda,t}\|^2 - \frac{\gamma}{8}\|\bar{D}_{\lambda,t}\|^2 + \frac{\gamma}{4}\|\bar{D}_{\lambda,t} - D_{\lambda,t}\|^2 + \frac{\gamma}{2}\left\|\nabla\phi(\lambda_t) - D^{\text{in}}_{\lambda,t}\right\|^2 + \frac{\gamma^2 L_\phi}{2}\|D_{\lambda,t}\|^2.\end{aligned} \quad (6.61)$$

where the last inequality follows from the inequality $-\|a\|^2 \leqslant -\frac{1}{2}\|b\|^2 + \|a - b\|^2$. □

### Lower level descent

We move now to descent properties on the lower variable $w_t$.

We start with a descent property that is shared with the dual variable $v_t$ in Section 6.B.1.3.

This follows because both updates—for $w_t$ and $v_t$—are gradient steps on strongly convex and smooth objectives.

**Lemma 6.12.** (Linear convergence of the lower and dual variables) *Assume that the lower and dual step sizes satisfy* $\tau, \rho \leqslant \frac{1}{L_{\mathcal{F},1}^{w,w}}$. *Then, for any* $t \geqslant 0$, *we have*

$$\begin{aligned} \|w_{t+1} - w_t^*\|^2 &\leqslant (1 - \tau\mu_{\mathcal{F}})\|w_t - w_t^*\|^2. \\ \|v_{t+1} - v^*(\lambda_t, w_t)\|^2 &\leqslant (1 - \rho\mu_{\mathcal{F}})\|v_t - v^*(\lambda_t, w_t)\|^2 \end{aligned} \tag{6.62}$$

*Proof.* For any $t \geqslant 0$,

$$\begin{aligned} \|w_{t+1} - w_t^*\|^2 &= \|w_t - \tau\nabla_w \mathcal{F}(w_t, \lambda_t) - w_t^*\|^2 \\ &= \|w_t - w_t^*\|^2 - \tau\langle w_t - w_t^*, \nabla_w \mathcal{F}(w_t, \lambda_t)\rangle \\ &\quad -\tau\langle w_t - w_t^*, \nabla_w \mathcal{F}(w_t, \lambda_t)\rangle + \tau^2\|\nabla_w \mathcal{F}(w_t, \lambda_t)\|^2. \end{aligned} \tag{6.63}$$

By strong convexity of $\mathcal{F}$ in $w$, $-\tau\langle w_t - w_t^*, \nabla_w \mathcal{F}(w_t, \lambda_t)\rangle \leqslant -\tau\mu_{\mathcal{F}}\|w_t - w_t^*\|^2$.

Furthermore, by cocoercivity of $\nabla_w \mathcal{F}$,

$$-\tau\langle w_t - w_t^*, \nabla_w \mathcal{F}(w_t, \lambda_t)\rangle + \tau^2\|\nabla_w \mathcal{F}(w_t, \lambda_t)\|^2 \leqslant -\tau\left(\frac{1}{L_{\mathcal{F},1}^{w,w}} - \tau\right)\|\nabla_w \mathcal{F}(w_t, \lambda_t)\|^2. \tag{6.64}$$

Thus,

$$\|w_{t+1} - w_t^*\|^2 \leqslant (1 - \tau\mu_{\mathcal{F}})\|w_t - w_t^*\|^2 - \tau\left(\frac{1}{L_{\mathcal{F},1}^{w,w}} - \tau\right)\|\nabla_w \mathcal{F}(w_t, \lambda_t)\|^2. \tag{6.65}$$

Finally, assuming $\tau \leqslant \frac{1}{L_{\mathcal{F},1}^{w,w}}$, the last term is nonpositive, so

$$\|w_{t+1} - w_t^*\|^2 \leqslant (1 - \tau\mu_{\mathcal{F}})\|w_t - w_t^*\|^2. \tag{6.66}$$

Finally, observe that the function $v \mapsto \frac{1}{2}v^\top \nabla_{w,w}^2 \mathcal{F}(w_t, \lambda_t)v + v^\top \nabla_w \mathcal{U}(w_t, \lambda_t)$ is $L_{\mathcal{F},1}^{w,w}$-smooth and $\mu_{\mathcal{F}}$-strongly convex by Assumption 6.5 (items 2 and 3). Thus, noting that the update on $v_t$ can be interpreted as a gradient step on this function and using the same rationale as above leads to

$$\|v_{t+1} - v^*(\lambda_t, w_t)\|^2 \leqslant (1 - \rho\mu_{\mathcal{F}})\|v_t - v^*(\lambda_t, w_t)\|^2, \tag{6.67}$$

for all $t \geqslant 0$. □

We next control the drift of the lower variable $w_t$ with respect to the implicit solution $w_t^*$.

**Lemma 6.13.** (Drift control of the lower variable) *Assume* $\tau \leqslant \frac{1}{L_{\mathcal{F},1}^{w,w}}$. *Then, for any* $t \geqslant 0$, *we have*

$$\left\|w_{t+1}-w_{t+1}^*\right\|^2 \leqslant \left(1-\frac{\tau\mu_{\mathcal{F}}}{2}\right)\|w_t-w_t^*\|^2+\frac{2}{\tau\mu_{\mathcal{F}}}\left(\frac{M_{\mathcal{C}}}{\mu_{\mathcal{F}}}\right)^2\gamma^2\left\|D_{\lambda,t}\right\|^2. \tag{6.68}$$

*Proof.* For any $t\in\mathbb{N}$, using Lemma 6.12 and Lemma 6.9, we have:

$$\begin{aligned}\left\|w_{t+1}-w_{t+1}^*\right\|^2 &= \left\|w_{t+1}-w_t^*+w_t^*-w_{t+1}^*\right\|^2\\ &= \|w_{t+1}-w_t^*\|^2+2\langle w_{t+1}-w_t^*,w_t^*-w_{t+1}^*\rangle+\left\|w_t^*-w_{t+1}^*\right\|^2\\ &\leqslant (1-\tau\mu_{\mathcal{F}})\|w_t-w_t^*\|^2+\left(\frac{M_{\mathcal{C}}}{\mu_{\mathcal{F}}}\right)^2\gamma^2\left\|D_{\lambda,t}\right\|^2+2\langle w_{t+1}-w_t^*,w_t^*-w_{t+1}^*\rangle.\end{aligned} \tag{6.69}$$

Using Young's inequality and Lemma 6.12, for any $\eta>0$, we obtain:

$$\begin{aligned}\langle w_{t+1}-w_t^*,w_t^*-w_{t+1}^*\rangle &\leqslant \frac{\eta}{2}\|w_{t+1}-w_t^*\|^2+\frac{1}{2\eta}\left\|w_t^*-w_{t+1}^*\right\|^2\\ &\leqslant \frac{\eta}{2}(1-\tau\mu_{\mathcal{F}})\|w_t-w_t^*\|^2+\frac{1}{2\eta}\left(\frac{M_{\mathcal{C}}}{\mu_{\mathcal{F}}}\right)^2\gamma^2\left\|D_{\lambda,t}\right\|^2.\end{aligned} \tag{6.70}$$

Taking $\eta=\frac{\tau\mu_{\mathcal{F}}}{2(1-\tau\mu_{\mathcal{F}})}$ finally gives

$$\begin{aligned}\left\|w_{t+1}-w_{t+1}^*\right\|^2 &\leqslant \left(1-\frac{\tau\mu_{\mathcal{F}}}{2}\right)\|w_t-w_t^*\|^2+\left(\frac{2}{\tau\mu_{\mathcal{F}}}-1\right)\left(\frac{M_{\mathcal{C}}}{\mu_{\mathcal{F}}}\right)^2\gamma^2\left\|D_{\lambda,t}\right\|^2\\ &\leqslant \left(1-\frac{\tau\mu_{\mathcal{F}}}{2}\right)\|w_t-w_t^*\|^2+\frac{2}{\tau\mu_{\mathcal{F}}}\left(\frac{M_{\mathcal{C}}}{\mu_{\mathcal{F}}}\right)^2\gamma^2\left\|D_{\lambda,t}\right\|^2.\end{aligned} \tag{6.71}$$

□

### Dual control

In this paragraph, we derive several bounds on the dual iterates $v_t$. We start with a boundedness result for the sequence $(v_t)_{t\geqslant 0}$. This result is crucial for controlling the error in the upper-level gradient estimate in Section 6.B.1.4, as it allows us to circumvent the issue of having a potentially unbounded mixed derivative $\nabla^2_{w,\lambda}\mathcal{F}$ for the lower-level objective.

In particular, prior analyses (Dagréou et al., 2022; Ghadimi & Wang, 2018) do not address these issues.

**Lemma 6.14.** (Boundedness of the dual variable) *Assume that the dual step size satisfies* $\rho\leqslant\frac{1}{L_{\mathcal{F},1}^{w,w}}$. *Then, the sequence* $(v_t)_{t\geqslant 0}$ *satisfies for all* $t\geqslant 0$,

$$\|v_t\|\leqslant\frac{4L_{\mathcal{U},0}}{\rho\mu_{\mathcal{F}}^2}+\epsilon_{v,0}, \tag{6.72}$$

*where* $\epsilon_{v,0}\triangleq\|v_0-v^*(\lambda_0,w_0)\|$.

*Proof.* We now prove by induction that for all $t \geqslant 0$,

$$\|v_t\| \leqslant \frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}} + \left[\sum_{s=1}^{t-1} (1-\rho\mu_{\mathcal{F}})^{\frac{s}{2}} \cdot \frac{2L_{\mathcal{U},0}}{\mu_{\mathcal{F}}}\right] + (1-\rho\mu_{\mathcal{F}})^{\frac{t}{2}} \epsilon_{v,0}. \tag{6.73}$$

First note that for all $t \geqslant 0$, $\|v^*(\lambda_t, w_t)\| = \left\|\nabla^2_{w,w}\mathcal{F}(w_t,\lambda_t)^{-1}\nabla\mathcal{U}(w_t,\lambda_t)\right\| \leqslant \frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}}$. Thus, for $t = 0$, we obtain,

$$\|v_1\| \leqslant \|v_1 - v^*(\lambda_0, w_0)\| + \|v^*(\lambda_0, w_0)\| \leqslant \epsilon_{v,0} + \frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}}, \tag{6.74}$$

which confirms the base case (using the convention that empty sums are zero). Assume now the result holds for some $t \geqslant 0$. Then, we have by Lemma 6.12

$$\begin{aligned}
\|v_{t+1}\| &\leqslant \|v_{t+1} - v^*(\lambda_t, w_t)\| + \|v^*(\lambda_t, w_t)\| \\
&\leqslant (1-\rho\mu_{\mathcal{F}})^{\frac{1}{2}} \|v_t - v^*(\lambda_t, w_t)\| + \|v^*(\lambda_t, w_t)\| \\
&\leqslant (1-\rho\mu_{\mathcal{F}})^{\frac{1}{2}} \|v_t\| + (1-\rho\mu_{\mathcal{F}})^{\frac{1}{2}} \frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}} + \frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}} \\
&\leqslant (1-\rho\mu_{\mathcal{F}})^{\frac{1}{2}} \left( \frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}} + \sum_{s=1}^{t-1} (1-\rho\mu_{\mathcal{F}})^{\frac{s}{2}} \cdot \frac{2L_{\mathcal{U},0}}{\mu_{\mathcal{F}}} \right. \\
&\qquad \left. + (1-\rho\mu_{\mathcal{F}})^{\frac{t}{2}} \epsilon_{v,0} \right) \\
&\quad + (1-\rho\mu_{\mathcal{F}})^{\frac{1}{2}} \frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}} + \frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}} \\
&= \frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}} + \sum_{s=1}^{t} (1-\rho\mu_{\mathcal{F}})^{\frac{s}{2}} \cdot \frac{2L_{\mathcal{U},0}}{\mu_{\mathcal{F}}} + (1-\rho\mu_{\mathcal{F}})^{\frac{t+1}{2}} \epsilon_{v,0},
\end{aligned} \tag{6.75}$$

This finishes the induction and we can conclude that for all $t \geqslant 0$,

$$\|v_t\| \leqslant \frac{2L_{\mathcal{U},0}}{\mu_{\mathcal{F}}} \sum_{t=0}^{\infty} (1-\rho\mu_{\mathcal{F}})^{\frac{t}{2}} + \epsilon_{v,0} = \frac{4L_{\mathcal{U},0}}{\rho\mu_{\mathcal{F}}^2} + \epsilon_{v,0}. \tag{6.76}$$

where we used the inequality $\sum_{t=0}^{\infty} (1-\alpha)^{\frac{t}{2}} \leqslant \frac{2}{\alpha}$ for all $\alpha \in (0,1]$. □

We now move to a bound on the drift of the dual solution $v^*(\lambda_t, w_t)$ along the iterates, which will be useful to control the dual error $\|v_t - v^*(\lambda_t, w_t)\|$ in the upcoming Lemma 6.17.

**Lemma 6.15.** (Drift of the dual solution) *For any $t \geqslant 0$, we have*

$$\|v^*(\lambda_{t+1}, w_{t+1}) - v^*(\lambda_t, w_t)\| \leqslant M_{v^*}\Big(\tau L_{\mathcal{F},1}^{w,w}\|w_t - w_t^*\| + \gamma\big\|D_{\lambda,t}\big\|\Big) \quad (6.77)$$

*where* $M_{v^*} \triangleq \left(L_{\mathcal{U},0}\frac{L_{\mathcal{F},2}^{w,w}}{\mu_{\mathcal{F}}^2} + \frac{L_{\mathcal{U},1}}{\mu_{\mathcal{F}}}\right)$.

*Proof.* Observe that,

$$\begin{aligned}\|v^*(\lambda_{t+1}, w_{t+1}) - v^*(\lambda_t, w_t)\| &\leqslant \left\|\nabla^2_{w,w}\mathcal{F}(w_{t+1},\lambda_{t+1})^{-1}\nabla\mathcal{U}(w_{t+1},\lambda_{t+1}) - \nabla^2_{w,w}\mathcal{F}(w_t,\lambda_t)^{-1}\nabla\mathcal{U}(w_t,\lambda_t)\right\| \\ &\leqslant \left\|\nabla^2_{w,w}\mathcal{F}(w_{t+1},\lambda_{t+1})^{-1} - \nabla^2_{w,w}\mathcal{F}(w_t,\lambda_t)^{-1}\right\|\cdot\|\nabla\mathcal{U}(w_{t+1},\lambda_{t+1})\| \\ &+\left\|\nabla^2_{w,w}\mathcal{F}(w_t,\lambda_t)^{-1}\right\|\cdot\|\nabla\mathcal{U}(w_{t+1},\lambda_{t+1}) - \nabla\mathcal{U}(w_t,\lambda_t)\|\end{aligned} \quad (6.78)$$

Therefore, using similar arguments as in the proof of Lemma 6.10, we obtain

$$\begin{aligned}\left\|\nabla^2_{w,w}\mathcal{F}(w_{t+1},\lambda_{t+1})^{-1} - \nabla^2_{w,w}\mathcal{F}(w_t,\lambda_t)^{-1}\right\| &\leqslant \frac{L_{\mathcal{F},2}^{w,w}}{\mu_{\mathcal{F}}^2}(\|w_{t+1} - w_t\| + \|\lambda_{t+1} - \lambda_t\|) \\ &= \frac{L_{\mathcal{F},2}^{w,w}}{\mu_{\mathcal{F}}^2}(\tau\|\nabla_w\mathcal{F}(w_t,\lambda_t)\| + \gamma\big\|D_{\lambda,t}\big\|) \\ &\leqslant \frac{L_{\mathcal{F},2}^{w,w}}{\mu_{\mathcal{F}}^2}\Big(\tau L_{\mathcal{F},1}^{w,w}\|w_t - w_t^*\| + \gamma\big\|D_{\lambda,t}\big\|\Big)\end{aligned} \quad (6.79)$$

where the last line follows from Assumption 6.5 (item 2). Besides, using Assumption 6.4 (item 2) and Assumption 6.5 (item 1), we have

$$\begin{aligned}\left\|\nabla^2_{w,w}\mathcal{F}(w_t,\lambda_t)^{-1}\right\|\cdot\|\nabla\mathcal{U}(w_{t+1},\lambda_{t+1}) - \nabla\mathcal{U}(w_t,\lambda_t)\| &\leqslant \frac{L_{\mathcal{U},1}}{\mu_{\mathcal{F}}}(\|w_{t+1} - w_t\| + \|\lambda_{t+1} - \lambda_t\|) \\ &\leqslant \frac{L_{\mathcal{U},1}}{\mu_{\mathcal{F}}}\Big(\tau L_{\mathcal{F},1}^{w,w}\|w_t - w_t^*\| + \gamma\big\|D_{\lambda,t}\big\|\Big)\end{aligned} \quad (6.80)$$

Hence, using one last time Assumption 6.4 (item 1), we get overall

$$\begin{aligned}\|v^*(\lambda_{t+1}, w_{t+1}) - v^*(\lambda_t, w_t)\| &\leqslant L_{\mathcal{U},0}\frac{L_{\mathcal{F},2}^{w,w}}{\mu_{\mathcal{F}}^2}\Big(\tau L_{\mathcal{F},1}^{w,w}\|w_t - w_t^*\| + \gamma\big\|D_{\lambda,t}\big\|\Big) \\ &+\frac{L_{\mathcal{U},1}}{\mu_{\mathcal{F}}}\Big(\tau L_{\mathcal{F},1}^{w,w}\|w_t - w_t^*\| + \gamma\big\|D_{\lambda,t}\big\|\Big) \\ &= M_{v^*}\Big(\tau L_{\mathcal{F},1}^{w,w}\|w_t - w_t^*\| + \gamma\big\|D_{\lambda,t}\big\|\Big)\end{aligned} \quad (6.81)$$

□

To get a bound on the dual error $\|v_t - v^*(\lambda_t, w_t)\|$, we must also control the partial error stemming from the drift of the lower variable $w_t$ with respect to the dual solution $\|v^*(\lambda_t, w_t) - v^*(\lambda_t)\|$.

**Lemma 6.16.** (Lower drift impact on the dual solution) *For any* $t \geqslant 0$*, we have*

$$\|v^*(\lambda_t, w_t) - v^*(\lambda_t)\| \leqslant M_{v^*}\|w_t - w_t^*\|. \tag{6.82}$$

*Proof.* Observe that

$$\begin{aligned}\|v^*(\lambda_t, w_t) - v^*(\lambda_t)\| &= \left\|\nabla^2_{w,w}\mathcal{F}(w_t, \lambda_t)^{-1}\nabla\mathcal{U}(w_t, \lambda_t) - \nabla^2_{w,w}\mathcal{F}(w_t^*, \lambda_t)^{-1}\nabla\mathcal{U}(w_t^*, \lambda_t)\right\| \\ &\leqslant \left\|\nabla^2_{w,w}\mathcal{F}(w_t, \lambda_t)^{-1} - \nabla^2_{w,w}\mathcal{F}(w_t^*, \lambda_t)^{-1}\right\| \cdot \|\nabla\mathcal{U}(w_t, \lambda_t)\| \\ &+ \left\|\nabla^2_{w,w}\mathcal{F}(w_t^*, \lambda_t)^{-1}\right\| \cdot \|\nabla\mathcal{U}(w_t, \lambda_t) - \nabla\mathcal{U}(w_t^*, \lambda_t)\|\end{aligned} \tag{6.83}$$

and using the same arguments as in the proof of Lemma 6.10, we obtain:

$$\begin{aligned}\left\|\nabla^2_{w,w}\mathcal{F}(w_t, \lambda_t)^{-1} - \nabla^2_{w,w}\mathcal{F}(w_t^*, \lambda_t)^{-1}\right\| \cdot \|\nabla\mathcal{U}(w_t, \lambda_t)\| &\leqslant \frac{L^{w,w}_{\mathcal{F},2}}{\mu^2_{\mathcal{F}}} \cdot L_{\mathcal{U},0}\|w_t - w_t^*\|, \\ \left\|\nabla^2_{w,w}\mathcal{F}(w_t^*, \lambda_t)^{-1}\right\| \cdot \|\nabla\mathcal{U}(w_t, \lambda_t) - \nabla\mathcal{U}(w_t^*, \lambda_t)\| &\leqslant \frac{L_{\mathcal{U},1}}{\mu_{\mathcal{F}}}\|w_t - w_t^*\|.\end{aligned} \tag{6.84}$$

Summing the two above inequalities gives the result. □

Our recursive bound on the dual error $\|v_t - v^*(\lambda_t)\|$ takes the following form.

**Lemma 6.17.** (Control of the dual error) *For any* $t \geqslant 0$, *we have*

$$\begin{aligned}\|v_{t+1} - v^*(\lambda_{t+1})\|^2 \leqslant \left(1 - \frac{\rho\mu_{\mathcal{F}}}{2}\right)\|v_t - v^*(\lambda_t)\|^2 + \frac{107}{\rho\mu_{\mathcal{F}}} \cdot M^2_{v^*}\|w_t - w_t^*\|^2 \\ + \frac{25}{\rho\mu_{\mathcal{F}}}M^2_{v^*}\left[\frac{2}{\tau\mu_{\mathcal{F}}}\left(\frac{M_{\mathcal{C}}}{\mu_{\mathcal{F}}}\right)^2 + 3\right]\gamma^2\|D_{\lambda,t}\|^2.\end{aligned} \tag{6.85}$$

*Proof.* For any $\alpha, \beta > 0$, we have by Young's inequality,

$$\begin{aligned}\|v_{t+1} - v^*(\lambda_{t+1})\|^2 &\leqslant (1+\alpha)\|v_{t+1} - v^*(\lambda_{t+1}, w_{t+1})\|^2 + \left(1 + \frac{1}{\alpha}\right)\|v^*(\lambda_{t+1}, w_{t+1}) - v^*(\lambda_{t+1})\|^2 \\ &\leqslant (1+\alpha)(1+\beta)\|v_{t+1} - v^*(\lambda_t, w_t)\|^2 \\ &+ (1+\alpha)\left(1 + \frac{1}{\beta}\right)\|v^*(\lambda_t, w_t) - v^*(\lambda_{t+1}, w_{t+1})\|^2 \\ &+ \left(1 + \frac{1}{\alpha}\right)\|v^*(\lambda_{t+1}, w_{t+1}) - v^*(\lambda_{t+1})\|^2\end{aligned} \tag{6.86}$$

Substituting Lemma 6.12, Lemma 6.15, and Lemma 6.16 into the preceding inequality yields

$$\begin{aligned}\|v_{t+1}-v^*(\lambda_{t+1})\|^2 \leqslant{}& (1+\alpha)(1+\beta)(1-\rho\mu_{\mathcal{F}})\|v_t-v^*(\lambda_t,w_t)\|^2\\ &+(1+\alpha)\left(1+\frac{1}{\beta}\right)2\tau^2\left(L_{\mathcal{F},1}^{w,w}\right)^2\left(L_{\mathcal{U},0}\frac{L_{\mathcal{F},2}^{w,w}}{\mu_{\mathcal{F}}^2}+\frac{L_{\mathcal{U},1}}{\mu_{\mathcal{F}}}\right)^2\|w_t-w_t^*\|^2\\ &+\left(1+\frac{1}{\alpha}\right)\left(L_{\mathcal{U},0}\frac{L_{\mathcal{F},2}^{w,w}}{\mu_{\mathcal{F}}^2}+\frac{L_{\mathcal{U},1}}{\mu_{\mathcal{F}}}\right)^2\|w_{t+1}-w_{t+1}^*\|^2\\ &+(1+\alpha)\left(1+\frac{1}{\beta}\right)2\gamma^2\left(L_{\mathcal{U},0}\frac{L_{\mathcal{F},2}^{w,w}}{\mu_{\mathcal{F}}^2}+\frac{L_{\mathcal{U},1}}{\mu_{\mathcal{F}}}\right)^2\|D_{\lambda,t}\|^2.\end{aligned}\tag{6.87}$$

Using once more Lemma 6.16 together with Young's inequality, we derive for any $\delta>0$

$$\|v_t-v^*(\lambda_t,w_t)\|^2\leqslant(1+\delta)\|v_t-v^*(\lambda_t)\|^2+\left(1+\frac{1}{\delta}\right)M_{v^*}^2\|w_t-w_t^*\|^2\tag{6.88}$$

Substituting the preceding inequality and Lemma 6.12 into (6.87) and rearranging yields:

$$\begin{aligned}\|v_{t+1}-v^*(\lambda_{t+1})\|^2 \leqslant{}& (1-\rho\mu_{\mathcal{F}})(1+\alpha)(1+\beta)(1+\delta)\|v_t-v^*(\lambda_t)\|^2\\ &+M_{v^*}^2\Bigg[(1-\rho\mu_{\mathcal{F}})(1+\alpha)(1+\beta)\left(1+\frac{1}{\delta}\right)\\ &+(1+\alpha)\left(1+\frac{1}{\beta}\right)\cdot 2\tau^2\left(L_{\mathcal{F},1}^{w,w}\right)^2\\ &+\left(1+\frac{1}{\alpha}\right)\left(1-\frac{\tau\mu_{\mathcal{F}}}{2}\right)\Bigg]\|w_t-w_t^*\|^2\\ &+M_{v^*}^2\Bigg[\left(1+\frac{1}{\alpha}\right)\frac{2}{\tau\mu_{\mathcal{F}}}\left(\frac{M_{\mathcal{C}}}{\mu_{\mathcal{F}}}\right)^2\\ &+2(1+\alpha)\left(1+\frac{1}{\beta}\right)\Bigg]\gamma^2\|D_{\lambda,t}\|^2.\end{aligned}\tag{6.89}$$

Finally, setting $\alpha=\beta=\delta=\frac{\rho\mu_{\mathcal{F}}}{24}$, we observe that $\alpha\leqslant\frac{\mu_{\mathcal{F}}}{24L_{\mathcal{F},1}^{w,w}}\leqslant 1$. This first ensures that $(1+\alpha)^3\leqslant 1+\alpha\sup_{1\leqslant s\leqslant 2}\frac{d}{dt}\left[t^3\right]\leqslant 1+12\alpha$,

which in turn implies

$$(1+\alpha)^3\cdot(1-\rho\mu_{\mathcal{F}})\leqslant(1+12\alpha)(1-\rho\mu_{\mathcal{F}})=\left(1+\frac{\rho\mu_{\mathcal{F}}}{2}\right)(1-\rho\mu_{\mathcal{F}})\leqslant\left(1-\frac{\rho\mu_{\mathcal{F}}}{2}\right).\tag{6.90}$$

And since $\rho\mu_{\mathcal{F}}\leqslant 1$, we have

$$\begin{aligned}\left(1+\frac{1}{\alpha}\right) &= 1+\frac{24}{\rho\mu_{\mathcal{F}}} \leqslant \frac{25}{\rho\mu_{\mathcal{F}}}\\ (1+\alpha)\cdot\left(1+\frac{1}{\alpha}\right) &\leqslant \left(\frac{25}{24}\right)\cdot\left(1+\frac{1}{\alpha}\right) \leqslant \frac{27}{\rho\mu_{\mathcal{F}}}\\ (1+\alpha)^2\cdot\left(1+\frac{1}{\alpha}\right) &\leqslant \left(\frac{25}{24}\right)^2\cdot\left(1+\frac{1}{\alpha}\right) \leqslant \frac{28}{\rho\mu_{\mathcal{F}}}\end{aligned} \tag{6.91}$$

This allows us to simplify (6.89) into

$$\begin{aligned}\|v_{t+1}-v^*(\lambda_{t+1})\|^2 \leqslant \left(1-\frac{\rho\mu_{\mathcal{F}}}{2}\right)&\|v_t-v^*(\lambda_t)\|^2\\ &+\frac{107}{\rho\mu_{\mathcal{F}}}\cdot M_{v^*}^2\|w_t-w_t^*\|^2\\ &+\frac{25}{\rho\mu_{\mathcal{F}}}M_{v^*}^2\left[\frac{2}{\tau\mu_{\mathcal{F}}}\left(\frac{M_{\mathcal{C}}}{\mu_{\mathcal{F}}}\right)^2+3\right]\gamma^2\left\|D_{\lambda,t}\right\|^2.\end{aligned} \tag{6.92}$$

where we also used the upper bounds $\tau \leqslant \frac{1}{L_{\mathcal{F},1}^{w,w}}$ and $\left(1-\tau\frac{\mu_{\mathcal{F}}}{2}\right) \leqslant 1$. □

### Control of the error on implicit gradient estimation

We now turn to the control of the error on the implicit gradient estimate $\left\|\nabla\phi(\lambda_t)-D_{\lambda,t}^{\text{in}}\right\|$.

This proposition will be combined with the descent Lemma 6.11 on the implicit objective to derive our Lyapunov analysis in the upcoming Section 6.B.1.5.

**Lemma 6.18.** (Error bound on the implicit gradient) *For any* $t \geqslant 0$*, we have*

$$\left\|\nabla\phi(\lambda_t)-D_{\lambda,t}^{\text{in}}\right\| \leqslant \left(L_{\mathcal{U},1}+L_{\mathcal{F},2}^{w,\lambda}\left(\frac{4L_{\mathcal{U},0}}{\rho\mu_{\mathcal{F}}^2}+\epsilon_{v,0}\right)\right)\|w_t-w_t^*\|+M_{\mathcal{C}}\|v^*(\lambda_t)-v_t\|. \tag{6.93}$$

*Proof.* For any $t \geqslant 0$, we have by the triangle inequality:

$$\begin{aligned}\left\|\nabla\phi(\lambda_t)-D_{\lambda,t}^{\text{in}}\right\| &= \left\|\nabla_\lambda\mathcal{U}(w_t^*,\lambda_t)+\nabla_{w,\lambda}^2\mathcal{F}(w_t^*,\lambda_t)^\top v^*(\lambda_t)-\left(\nabla_\lambda\mathcal{U}(w_t,\lambda_t)+\nabla_{w,\lambda}^2\mathcal{F}(w_t,\lambda_t)^\top v_t\right)\right\|\\ &\leqslant \|\nabla_\lambda\mathcal{U}(w_t^*,\lambda_t)-\nabla_\lambda\mathcal{U}(w_t,\lambda_t)\|+\left\|\nabla_{w,\lambda}^2\mathcal{F}(w_t^*,\lambda_t)^\top v^*(\lambda_t)-\nabla_{w,\lambda}^2\mathcal{F}(w_t,\lambda_t)^\top v_t\right\|\end{aligned} \tag{6.94}$$

The first term can be upper bounded using Assumption 6.4 (item 2): $\|\nabla_\lambda\mathcal{U}(w_t^*,\lambda_t)-\nabla_\lambda\mathcal{U}(w_t,\lambda_t)\| \leqslant L_{\mathcal{U},1}\|w_t-w_t^*\|$.

Now consider the second term. First observe that

$$\begin{aligned}&\left\|\nabla^2_{w,\lambda}\mathcal{F}(w_t^*,\lambda_t)v^*(\lambda_t)-\nabla^2_{w,\lambda}\mathcal{F}(w_t,\lambda_t)v_t\right\|\\&\leqslant\left\|\nabla^2_{w,\lambda}\mathcal{F}(w_t^*,\lambda_t)\right\|\cdot\|v^*(\lambda_t)-v_t\|\\&\quad+\left\|\nabla^2_{w,\lambda}\mathcal{F}(w_t^*,\lambda_t)-\nabla^2_{w,\lambda}\mathcal{F}(w_t,\lambda_t)\right\|\cdot\|v_t\|\end{aligned}\tag{6.95}$$

By Assumption 6.5 (items 4 and 5), we note that $\left\|\nabla^2_{w,\lambda}\mathcal{F}(w_t^*,\lambda_t)\right\|\leqslant M_{\mathcal{C}}$, and $\left\|\nabla^2_{w,\lambda}\mathcal{F}(w_t^*,\lambda_t)-\nabla^2_{w,\lambda}\mathcal{F}(w_t,\lambda_t)\right\|\leqslant L^{w,\lambda}_{\mathcal{F},2}\|w_t-w_t^*\|$. Furthermore, Lemma 6.14 gives $\|v_t\|\leqslant\frac{4L_{\mathcal{U},0}}{\rho\mu^2_{\mathcal{F}}}+\epsilon_{v,0}$, so that

$$\left\|\nabla^2_{w,\lambda}\mathcal{F}(w_t^*,\lambda_t)v^*(\lambda_t)-\nabla^2_{w,\lambda}\mathcal{F}(w_t,\lambda_t)v_t\right\|\leqslant M_{\mathcal{C}}\|v^*(\lambda_t)-v_t\|+L^{w,\lambda}_{\mathcal{F},2}\left(\frac{4L_{\mathcal{U},0}}{\rho\mu^2_{\mathcal{F}}}+\epsilon_{v,0}\right)\|w_t-w_t^*\|.\tag{6.96}$$

Hence, we have overall

$$\left\|\nabla\phi(\lambda_t)-D^{\text{in}}_{\lambda,t}\right\|\leqslant\left(L_{\mathcal{U},1}+L^{w,\lambda}_{\mathcal{F},2}\left(\frac{4L_{\mathcal{U},0}}{\rho\mu^2_{\mathcal{F}}}+\epsilon_{v,0}\right)\right)\|w_t-w_t^*\|+M_{\mathcal{C}}\|v^*(\lambda_t)-v_t\|.\tag{6.97}$$

□

As an immediate consequence of the preceding lemma and the non-expansiveness of the projection operator on the convex set $\mathcal{C}$, we have the following corollary.

**Corollary 6.19.** (Error bound on the generalized gradient) *For any* $t\geqslant 0$*, we have*

$$\|D_{\lambda,t}-\bar{D}_{\lambda,t}\|\leqslant\left(L_{\mathcal{U},1}+L^{w,\lambda}_{\mathcal{F},2}\left(\frac{4L_{\mathcal{U},0}}{\rho\mu^2_{\mathcal{F}}}+\epsilon_{v,0}\right)\right)\|w_t-w_t^*\|+M_{\mathcal{C}}\|v^*(\lambda_t)-v_t\|.\tag{6.98}$$

Proof of the deterministic convergence theorem

*Proof.* Let us introduce the Lyapunov function $\psi_t=\phi(\lambda_t)+\alpha\|w_t-w_t^*\|^2+\beta\|v_t-v^*(\lambda_t)\|^2$ for some $\alpha,\beta>0$. By Lemma 6.26, setting $\alpha$ and $\beta$ as

$$\begin{aligned}\alpha&=6\gamma\kappa_{\mathcal{F},w}\left(L_{\mathcal{U},1}+L^{w,\lambda}_{\mathcal{F},2}\left(4\kappa_{\mathcal{F},w}\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}}+\epsilon_{v,0}\right)\right)^2+1284\gamma\kappa^3_{\mathcal{F},w}M^2_{\mathcal{C}}M^2_{v^*}\\\beta&=3\gamma\kappa_{\mathcal{F},w}M^2_{\mathcal{C}},\end{aligned}\tag{6.99}$$

together with the parameterization $\tau=\rho=\frac{1}{L^{w,w}_{\mathcal{F},1}}$, and $\gamma\leqslant\min\left(\frac{1}{\kappa_{\mathcal{F},w}},\bar{\gamma}\right)$ where

$$\bar{\gamma}^{-1} = 2(L_\phi) + 48\kappa_{\mathcal{F},w}\left(\frac{M_{\mathcal{C}}}{\mu_{\mathcal{F}}}\right)^2\left(L_{\mathcal{U},1} + L^{w,\lambda}_{\mathcal{F},2}\left(4\kappa_{\mathcal{F},w}\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}} + \epsilon_{v,0}\right)\right)^2 + 10272\kappa^3_{\mathcal{F},w}\frac{M^4_{\mathcal{C}}}{\mu_{\mathcal{F}^2}}M^2_{v^*} + 300\kappa_{\mathcal{F},w}M^2_{\mathcal{C}}M^2_{v^*}\left[2\kappa_{\mathcal{F},w}\left(\frac{M_{\mathcal{C}}}{\mu_{\mathcal{F}}}\right)^2 + 3\right]. \tag{6.100}$$

ensures that $\psi_{t+1} \leqslant \psi_t - \frac{\gamma}{8}\left\|\bar{D}_{\lambda,t}\right\|^2 - \frac{\alpha}{4\kappa_{\mathcal{F},w}}\|w_t - w^*_t\|^2$ for all $t \geqslant 0$. Summing this inequality for $t = 0, ..., T-1$ gives

$$\frac{1}{T}\sum_{t=0}^{T-1}\left[\left\|\bar{D}_{\lambda,t}\right\|^2 + \frac{2\alpha}{\gamma\kappa_{\mathcal{F},w}}\|w_t - w^*_t\|^2\right] \leqslant \frac{8}{\gamma T}(\psi_0 - \psi_T) \leqslant \frac{8}{\gamma T}\Big(\phi(\lambda_0) - \inf\phi + \alpha\epsilon^2_{w,0} + \beta\epsilon^2_{v,0}\Big). \tag{6.101}$$

Setting $\bar{\alpha} = 2\frac{\alpha}{\kappa_{\mathcal{F},w}\gamma}$ and $\bar{\beta} = 2\frac{\beta}{\kappa_{\mathcal{F},w}\gamma}$ concludes the proof. □

### 6.B.2 The stochastic case

We now turn to the proof of Theorem 6.8.

To this end, we introduce the stochastic gradient-estimation errors $\Delta^w_{\mathcal{U},1}$, $\Delta^\lambda_{\mathcal{U},1}$, $\Delta_{\mathcal{F},1}$, $\Delta^{w,w}_{\mathcal{F},2}$, and $\Delta^{w,\lambda}_{\mathcal{F},2}$ as follows:

$$\begin{aligned}
\Delta^w_{\mathcal{U},1,t} &= \tilde{\nabla}_w\mathcal{U}(w_t,\lambda_t) - \nabla_w\mathcal{U}(w_t,\lambda_t) & \Delta_{\mathcal{F},1,t} &= \tilde{\nabla}_w\mathcal{F}(w_t,\lambda_t) - \nabla_w\mathcal{F}(w_t,\lambda_t)\\
\Delta^\lambda_{\mathcal{U},1,t} &= \tilde{\nabla}_\lambda\mathcal{U}(w_t,\lambda_t) - \nabla_\lambda\mathcal{U}(w_t,\lambda_t) & \Delta^{w,w}_{\mathcal{F},2,t} &= \tilde{\nabla}^2_{w,w}\mathcal{F}(w_t,\lambda_t) - \nabla^2_{w,w}\mathcal{F}(w_t,\lambda_t)\\
& & \Delta^{w,\lambda}_{\mathcal{F},2,t} &= \tilde{\nabla}^2_{w,\lambda}\mathcal{F}(w_t,\lambda_t) - \nabla^2_{w,\lambda}\mathcal{F}(w_t,\lambda_t),
\end{aligned} \tag{6.102}$$

We now use the notation $D_{\lambda,t}$, $D^{in}_{\lambda,t}$, and $\bar{D}_{\lambda,t}$ as follows:

$$\begin{aligned}
D_{\lambda,t} &\triangleq \frac{1}{\gamma_t}\Big(\lambda_t - \mathrm{Proj}_{\mathcal{C}}\Big(\lambda_t - \gamma_t D^{in}_{\lambda,t}\Big)\Big)\\
D^{in}_{\lambda,t} &\triangleq \tilde{\nabla}_\lambda\mathcal{U}(w_t,\lambda_t) + \tilde{\nabla}^2_{w,\lambda}\mathcal{F}(w_t,\lambda_t)^\top v_t,\\
\bar{D}_{\lambda,t} &\triangleq \frac{1}{\gamma_t}(\lambda_t - \mathrm{Proj}_{\mathcal{C}}(\lambda_t - \gamma_t\nabla\phi(\lambda_t)))
\end{aligned} \tag{6.103}$$

for the stochastic counterparts of the approximate gradient, the approximate projected gradient, and the exact projected gradient introduced in the last section.

#### Expected lower-level descent

We first provide stochastic counterparts to the descent properties derived in Section 6.B.1.2.

#### Proof for expected lower-level descent

**Lemma 6.20.** (Stochastic descent of the lower, dual and upper variables) *Assume that the lower and dual step sizes satisfy* $\tau, \rho \leqslant 1/L_{\mathcal{F},1}^{w,w}$. *Then, for any* $t \geqslant 0$, *we have*

$$\begin{aligned}
\mathbb{E}\Big[\|w_{t+1} - w_t^*\|^2 \mid \mathfrak{S}_t\Big] &\leqslant (1 - \tau\mu_{\mathcal{F}})\|w_t - w_t^*\|^2 + \tau^2 \mathcal{V}_{\mathcal{F},1} \\
\mathbb{E}\Big[\|v_{t+1} - v^*(\lambda_t, w_t)\|^2 \mid \mathfrak{S}_t\Big] &\leqslant (1 - \rho\mu_{\mathcal{F}})\|v_t - v^*(\lambda_t, w_t)\|^2 + \rho^2 \mathcal{V}_{\mathcal{F},2}^{w,w}\|v_t\|^2 + \rho^2 \mathcal{V}_{\mathcal{U},1}^{w} \\
\mathbb{E}\Big[\|w_{t+1} - w_{t+1}^*\|^2 \mid \mathfrak{S}_t\Big] &\leqslant \Big(1 - \frac{\tau\mu_{\mathcal{F}}}{2}\Big)\|w_t - w_t^*\|^2 + \frac{3}{\tau\mu_{\mathcal{F}}}\Big(\frac{M_{\mathcal{C}}}{\mu_{\mathcal{F}}}\Big)^2 \mathbb{E}\Big[\|\gamma_t D_{\lambda,t}\|^2 \mid \mathfrak{S}_t\Big] + \frac{\tau^2 \kappa_{\mathcal{F},w}}{\kappa_{\mathcal{F},w} - 1}\mathcal{V}_{\mathcal{F},1}.
\end{aligned} \tag{6.104}$$

*Proof.* For any $t \geqslant 0$, we have

$$\begin{aligned}
\|w_{t+1} - w_t^*\|^2 &= \Big\|w_t - \tau\tilde{\nabla}_w \mathcal{F}(w_t, \lambda_t) - w_t^*\Big\|^2 \\
&= \|w_t - \tau\nabla_w \mathcal{F}(w_t, \lambda_t) - w_t^* - \tau\Delta_{\mathcal{F},1,t}\|^2 \\
&= \|w_t - \tau\nabla_w \mathcal{F}(w_t, \lambda_t) - w_t^*\|^2 - 2\tau\langle w_t - \tau\nabla_w \mathcal{F}(w_t, \lambda_t) - w_t^*, \Delta_{\mathcal{F},1,t}\rangle + \tau^2\|\Delta_{\mathcal{F},1,t}\|^2.
\end{aligned} \tag{6.105}$$

Taking the expectation with respect to the stochasticity at iteration $t$ gives

$$\begin{aligned}
\mathbb{E}\Big[\|w_{t+1} - w_t^*\|^2 \mid \mathfrak{S}_t\Big] &\leqslant \|w_t - \tau\nabla_w \mathcal{F}(w_t, \lambda_t) - w_t^*\|^2 + \tau^2 \mathcal{V}_{\mathcal{F},1} \\
&\leqslant (1 - \tau\mu_{\mathcal{F}})\|w_t - w_t^*\|^2 + \tau^2 \mathcal{V}_{\mathcal{F},1}
\end{aligned} \tag{6.106}$$

where the last inequality follows from the analysis of the deterministic case, assuming $\tau \leqslant 1/L_{\mathcal{F},1}^{w,w}$. Similarly, we have for all $t \geqslant 0$,

$$\begin{aligned}
\|v_{t+1} - v^*(\lambda_t, w_t)\|^2 &= \Big\|v_t - \rho\Big(\tilde{\nabla}_w^2 \mathcal{F}(w_t, \lambda_t)v_t + \tilde{\nabla}_w \mathcal{U}(w_t, \lambda_t)\Big) - v^*(\lambda_t, w_t)\Big\|^2 \\
&= \|v_t - \rho\big(\nabla_w^2 \mathcal{F}(w_t, \lambda_t)v_t + \nabla_w \mathcal{U}(w_t, \lambda_t)\big) - v^*(\lambda_t, w_t)\|^2 \\
&\quad + 2\rho\langle v_t - \rho\big(\nabla_w^2 \mathcal{F}(w_t, \lambda_t)v_t + \nabla_w \mathcal{U}(w_t, \lambda_t)\big) - v^*(\lambda_t, w_t), \rho\Big(\Delta_{\mathcal{F},2,t}^{w,w}v_t + \Delta_{\mathcal{U},1,t}\Big)\rangle \\
&\quad + \rho^2\Big\|\Delta_{\mathcal{F},2,t}^{w,w}v_t + \Delta_{\mathcal{U},1,t}\Big\|^2.
\end{aligned} \tag{6.107}$$

and using the linearity of the conditional expectation, and independence between the two random estimates $\Delta_{\mathcal{F},2,t}^{w,w}$ and $\Delta_{\mathcal{U},1,t}$, we get

$$\begin{aligned}
\mathbb{E}\Big[\|v_{t+1} - v^*(\lambda_t, w_t)\|^2 \mid \mathfrak{S}_t\Big] &\leqslant \|v_t - \rho\big(\nabla_w^2 \mathcal{F}(w_t, \lambda_t)v_t + \nabla_w \mathcal{U}(w_t, \lambda_t)\big) - v^*(\lambda_t, w_t)\|^2 \\
&\quad + \rho^2\Big(\mathcal{V}_{\mathcal{F},2}^{w,w}\|v_t\|^2 + \mathcal{V}_{\mathcal{U},1}^{w}\Big) \\
&\leqslant (1 - \rho\mu_{\mathcal{F}})\|v_t - v^*(\lambda_t, w_t)\|^2 + \rho^2\Big(\mathcal{V}_{\mathcal{F},2}^{w,w}\|v_t\|^2 + \mathcal{V}_{\mathcal{U},1}^{w}\Big)
\end{aligned} \tag{6.108}$$

where the last inequality also follows from the deterministic descent analysis and having $\rho \leqslant 1/L_{\mathcal{F},1}^{w,w}$. Using now the same rationale as in the deterministic lower-level drift analysis, we have for any $\eta > 0$,

$$\begin{aligned}\mathbb{E}\Big[\|w_{t+1}-w^*_{t+1}\|^2 \mid \mathfrak{S}_t\Big] &\leqslant (1+\eta)\mathbb{E}\Big[\|w_{t+1}-w^*_t\|^2 \mid \mathfrak{S}_t\Big] + (1+\eta^{-1})\mathbb{E}\Big[\|w^*_t-w^*_{t+1}\|^2 \mid \mathfrak{S}_t\Big] \\ &\leqslant (1+\eta)\Big((1-\tau\mu_{\mathcal{F}})\|w_t-w^*_t\|^2+\tau^2\mathcal{V}_{\mathcal{F},1}\Big) + (1+\eta^{-1})\Big(\frac{M_{\mathcal{C}}}{\mu_{\mathcal{F}}}\Big)^2\mathbb{E}\Big[\|\gamma_t D_{\lambda,t}\|^2 \mid \mathfrak{S}_t\Big]\end{aligned} \tag{6.109}$$

and setting again $\eta = \frac{\tau\mu_{\mathcal{F}}}{2(1-\tau\mu_{\mathcal{F}})}$ gives

$$\begin{aligned}\mathbb{E}\Big[\|w_{t+1}-w^*_{t+1}\|^2 \mid \mathfrak{S}_t\Big] &\leqslant \Big(1-\frac{\tau\mu_{\mathcal{F}}}{2}\Big)\|w_t-w^*_t\|^2 + \Big(\frac{M_{\mathcal{C}}}{\mu_{\mathcal{F}}}\Big)^2\Big(1+\frac{2(1-\tau\mu_{\mathcal{F}})}{\tau\mu_{\mathcal{F}}}\Big)\mathbb{E}\Big[\|\gamma_t D_{\lambda,t}\|^2 \mid \mathfrak{S}_t\Big] \\ &\quad +\tau^2\Big(1+\frac{\tau\mu_{\mathcal{F}}}{2(1-\tau\mu_{\mathcal{F}})}\Big)\mathcal{V}_{\mathcal{F},1} \\ &\leqslant \Big(1-\frac{\tau\mu_{\mathcal{F}}}{2}\Big)\|w_t-w^*_t\|^2 + \frac{3}{\tau\mu_{\mathcal{F}}}\Big(\frac{M_{\mathcal{C}}}{\mu_{\mathcal{F}}}\Big)^2\mathbb{E}\Big[\|\gamma_t D_{\lambda,t}\|^2 \mid \mathfrak{S}_t\Big] + \tau^2\frac{\kappa_{\mathcal{F},w}}{\kappa_{\mathcal{F},w}-1}\mathcal{V}_{\mathcal{F},1}\end{aligned} \tag{6.110}$$

where the last line follows from having $\tau \leqslant 1/L^{w,w}_{\mathcal{F},1} \leqslant 1/\mu_{\mathcal{F}}$ and $\kappa_{\mathcal{F},w} \triangleq L^{w,w}_{\mathcal{F},1}/\mu_{\mathcal{F}} \geqslant 1$. □

### Proofs of the stochastic dual-control lemmas

**Lemma 6.21.** (Expected boundedness of the dual variable) *Assume that the dual step size satisfies* $\rho \leqslant \min\Big(1/L^{w,w}_{\mathcal{F},1}, \mu_{\mathcal{F}}/4\mathcal{V}^{w,w}_{\mathcal{F},2}\Big)$. *Then, for any* $t \geqslant 0$, *we have*

$$\begin{aligned}\mathbb{E}\Big[\|\nu_{t+1}\|^2 \mid \mathfrak{S}_t\Big] &\leqslant (1-\rho\mu_{\mathcal{F}}/8)\|\nu_t\|^2 + \frac{15}{\rho\mu_{\mathcal{F}}}\Big(\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}}\Big)^2 + 2\rho^2\mathcal{V}^w_{\mathcal{U},1} \\ \mathbb{E}\Big[\|\nu_t\|^2\Big] &\leqslant \mathcal{C}_{\nu,*}\end{aligned} \tag{6.111}$$

*where* $\mathcal{C}_{\nu,*} \triangleq 2\epsilon_{\nu,0} + \frac{121}{\rho^2\mu^2_{\mathcal{F}}}\Big(\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}}\Big)^2 + \frac{16\rho}{\mu_{\mathcal{F}}}\mathcal{V}^w_{\mathcal{U},1}$.

*Proof.* By Lemma 6.20 and Young's inequality, we have for any $\rho \leqslant 1/L^{w,w}_{\mathcal{F},1}$ and $\eta > 0$,

$$\begin{aligned}\mathbb{E}\Big[\|\nu_{t+1}-\nu^*(\lambda_t,w_t)\|^2 \mid \mathfrak{S}_t\Big] &\leqslant (1-\rho\mu_{\mathcal{F}})\|\nu_t-\nu^*(\lambda_t,w_t)\|^2 + \rho^2\Big(\mathcal{V}^{w,w}_{\mathcal{F},2}\|\nu_t\|^2 + \mathcal{V}^w_{\mathcal{U},1}\Big) \\ &\leqslant \Big((1+\eta)-\rho\Big(\mu_{\mathcal{F}}(1+\eta)-\rho\mathcal{V}^{w,w}_{\mathcal{F},2}\Big)\Big)\|\nu_t\|^2 \\ &\quad +(1-\rho\mu_{\mathcal{F}})(1+1/\eta)\|\nu^*(\lambda_t,w_t)\|^2 + \rho^2\mathcal{V}^w_{\mathcal{U},1}.\end{aligned} \tag{6.112}$$

Setting $\eta = \rho\mu_{\mathcal{F}}/2$ and having $\rho \leqslant \mu_{\mathcal{F}}/\Big(4\mathcal{V}^{w,w}_{\mathcal{F},2}\Big)$ ensures that

$$\Big((1+\eta)-\rho\Big(\mu_{\mathcal{F}}(1+\eta)-\rho\mathcal{V}^{w,w}_{\mathcal{F},2}\Big)\Big) \leqslant 1-\rho\mu_{\mathcal{F}}/4. \tag{6.113}$$

This in turn gives

$$\mathbb{E}\Big[\|\nu_{t+1}-\nu^*(\lambda_t,w_t)\|^2 \mid \mathfrak{S}_t\Big] \leqslant (1-\rho\mu_{\mathcal{F}}/4)\|\nu_t\|^2 + \frac{3}{\rho\mu_{\mathcal{F}}}\Big(\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}}\Big)^2 + \rho^2\mathcal{V}^w_{\mathcal{U},1}. \tag{6.114}$$

where we used the upper bound on $\|v^*(\lambda_0, w_0)\|^2$ given in the deterministic dual-boundedness analysis and that $0 \leqslant \rho\mu_{\mathcal{F}} \leqslant 1$. Hence, using one more time Young's inequality, we obtain for any $\nu > 0$,

$$\begin{aligned}\mathbb{E}\Big[\|v_{t+1}\|^2 \mid \mathfrak{S}_t\Big] &\leqslant (1+\nu)\mathbb{E}\Big[\|v_{t+1} - v^*(\lambda_t, w_t)\|^2 \mid \mathfrak{S}_t\Big] + (1+1/\nu)\|v^*(\lambda_t, w_t)\|^2 \\ &\leqslant (1+\nu)(1-\rho\mu_{\mathcal{F}}/4)\|v_t\|^2 + (1+\nu)\left(\frac{3}{\rho\mu_{\mathcal{F}}}\left(\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}}\right)^2 + \rho^2\mathcal{V}^w_{\mathcal{U},1}\right) + (1+1/\nu)\left(\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}}\right)^2. \end{aligned} \tag{6.115}$$

Setting now $\nu = \rho\mu_{\mathcal{F}}/8$ gives $(1+\nu)(1-\rho\mu_{\mathcal{F}}/4) \leqslant 1 - \rho\mu_{\mathcal{F}}/8$ and we obtain

$$\mathbb{E}\Big[\|v_{t+1}\|^2 \mid \mathfrak{S}_t\Big] \leqslant (1-\rho\mu_{\mathcal{F}}/8)\|v_t\|^2 + \frac{15}{\rho\mu_{\mathcal{F}}}\left(\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}}\right)^2 + 2\rho^2\mathcal{V}^w_{\mathcal{U},1} \tag{6.116}$$

where we used the upper bounds $1+\nu \leqslant 2, 1+1/\nu \leqslant 9/(\rho\mu_{\mathcal{F}})$. Using the tower property of conditional expectation, we can iterate the above recursion down to $t = 0$ and obtain

$$\begin{aligned}\mathbb{E}\Big[\|v_t\|^2\Big] &\leqslant (1-\rho\mu_{\mathcal{F}}/8)^t\|v_0\|^2 + \left(\frac{15}{\rho\mu_{\mathcal{F}}}\left(\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}}\right)^2 + 2\rho^2\mathcal{V}^w_{\mathcal{U},1}\right)\sum_{s=0}^{t-1}(1-\rho\mu_{\mathcal{F}}/8)^s \\ &\leqslant (1-\rho\mu_{\mathcal{F}}/8)^t\|v_0\|^2 + \frac{120}{\rho^2\mu_{\mathcal{F}}^2}\left(\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}}\right)^2 + \frac{16\rho}{\mu_{\mathcal{F}}}\mathcal{V}^w_{\mathcal{U},1} \\ &\leqslant 2\epsilon_{v,0} + \frac{121}{\rho^2\mu_{\mathcal{F}}^2}\left(\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}}\right)^2 + \frac{16\rho}{\mu_{\mathcal{F}}}\mathcal{V}^w_{\mathcal{U},1}\end{aligned} \tag{6.117}$$

where the last line again uses Young's inequality and the fact that $0 \leqslant \rho\mu_{\mathcal{F}} \leqslant 1$. □

**Lemma 6.22.** (Stochastic drift of the dual solution) *For any $t \geqslant 0$, we have*

$$\begin{aligned}&\mathbb{E}\Big[\|v^*(\lambda_{t+1}, w_{t+1}) - v^*(\lambda_t, w_t)\|^2 \mid \mathfrak{S}_t\Big] \leqslant L_{v,*}\left(\tau^2\left(L^{w,w}_{\mathcal{F},1}\right)^2\|w_t - w_t^*\|^2 + \tau^2\mathcal{V}_{\mathcal{F},1} + \mathbb{E}\Big[\|\gamma_t D_{\lambda,t}\|^2 \mid \mathfrak{S}_t\Big]\right), \\ &\|v^*(\lambda_t, w_t) - v^*(\lambda_t)\|^2 \leqslant L_{v,*}\|w_t - w_t^*\|^2\end{aligned} \tag{6.118}$$

*where* $L_{v,*} \triangleq 2\mu_{\mathcal{F}}^{-2}\left(\left(L_{\mathcal{U},0}\cdot L^{w,w}_{\mathcal{F},2}/\mu_{\mathcal{F}}\right)^2 + \left(L_{\mathcal{U},1}\right)^2\right)$.

*Proof.* For any $t \geqslant 0$, using the same argument as in the deterministic drift analysis, we have

$$\begin{aligned}\left\|\nabla^2_{w,w}\mathcal{F}(w_{t+1}, \lambda_{t+1})^{-1} - \nabla^2_{w,w}\mathcal{F}(w_t, \lambda_t)^{-1}\right\|^2 &\leqslant \left(\frac{L^{w,w}_{\mathcal{F},2}}{\mu^2_{\mathcal{F}}}\right)^2\left(\|w_{t+1} - w_t\|^2 + \|\lambda_{t+1} - \lambda_t\|^2\right) \\ &= \left(\frac{L^{w,w}_{\mathcal{F},2}}{\mu^2_{\mathcal{F}}}\right)^2\left(\tau^2\left\|\tilde{\nabla}_w\mathcal{F}(w_t, \lambda_t)\right\|^2 + \|\gamma_t D_{\lambda,t}\|^2\right).\end{aligned} \tag{6.119}$$

Hence, using again Assumption 6.4 (item 1) and taking the expectation with respect to $\mathfrak{S}_t$, we obtain

$$\begin{aligned}
&\mathbb{E}\left[\left\|\nabla^2_{w,w}\mathcal{F}(w_{t+1},\lambda_{t+1})^{-1}-\nabla^2_{w,w}\mathcal{F}(w_t,\lambda_t)^{-1}\right\|^2\cdot\|\nabla_w\mathcal{U}(w_{t+1},\lambda_{t+1})\|^2\mid\mathfrak{S}_t\right]\\
&\quad\leqslant\left(L_{\mathcal{U},0}\right)^2\left(\frac{L^{w,w}_{\mathcal{F},2}}{\mu^2_{\mathcal{F}}}\right)^2\left(\tau^2\|\nabla_w\mathcal{F}(w_t,\lambda_t)\|^2+\tau^2\mathcal{V}_{\mathcal{F},1}+\mathbb{E}\left[\left\|\gamma_t D_{\lambda,t}\right\|^2\mid\mathfrak{S}_t\right]\right)\\
&\quad\leqslant\left(L_{\mathcal{U},0}\right)^2\left(\frac{L^{w,w}_{\mathcal{F},2}}{\mu^2_{\mathcal{F}}}\right)^2\left(\tau^2\left(L^{w,w}_{\mathcal{F},1}\right)^2\|w_t-w_t^*\|^2+\tau^2\mathcal{V}_{\mathcal{F},1}+\mathbb{E}\left[\left\|\gamma_t D_{\lambda,t}\right\|^2\mid\mathfrak{S}_t\right]\right).
\end{aligned}\tag{6.120}$$

Similarly, for any $t\geqslant 0$, we obtain

$$\begin{aligned}
&\mathbb{E}\left[\left\|\nabla^2_{w,w}\mathcal{F}(w_t,\lambda_t)^{-1}\right\|^2\cdot\|\nabla_w\mathcal{U}(w_{t+1},\lambda_{t+1})-\nabla_w\mathcal{U}(w_t,\lambda_t)\|^2\mid\mathfrak{S}_t\right]\\
&\quad\leqslant\left(\frac{L_{\mathcal{U},1}}{\mu_{\mathcal{F}}}\right)^2\left(\tau^2\left(L^{w,w}_{\mathcal{F},1}\right)^2\|w_t-w_t^*\|^2+\tau^2\mathcal{V}_{\mathcal{F},1}+\mathbb{E}\left[\left\|\gamma_t D_{\lambda,t}\right\|^2\mid\mathfrak{S}_t\right]\right).
\end{aligned}\tag{6.121}$$

Hence, using the convexity of $\|\cdot\|^2$ and combining the two previous bounds, we obtain

$$\begin{aligned}
&\mathbb{E}\left[\|v^*(\lambda_{t+1},w_{t+1})-v^*(\lambda_t,w_t)\|^2\mid\mathfrak{S}_t\right]\\
&\quad\leqslant 2\mathbb{E}\left[\left\|\nabla^2_{w,w}\mathcal{F}(w_{t+1},\lambda_{t+1})^{-1}-\nabla^2_{w,w}\mathcal{F}(w_t,\lambda_t)^{-1}\right\|^2\cdot\|\nabla_w\mathcal{U}(w_{t+1},\lambda_{t+1})\|^2\mid\mathfrak{S}_t\right]\\
&\qquad+2\mathbb{E}\left[\left\|\nabla^2_{w,w}\mathcal{F}(w_t,\lambda_t)^{-1}\right\|^2\cdot\|\nabla_w\mathcal{U}(w_{t+1},\lambda_{t+1})-\nabla_w\mathcal{U}(w_t,\lambda_t)\|^2\mid\mathfrak{S}_t\right]\\
&\quad\leqslant 2\mu_{\mathcal{F}}^{-2}\left(\left(L_{\mathcal{U},0}\cdot L^{w,w}_{\mathcal{F},2}/\mu_{\mathcal{F}}\right)^2+\left(L_{\mathcal{U},1}\right)^2\right)\left(\tau^2\left(L^{w,w}_{\mathcal{F},1}\right)^2\|w_t-w_t^*\|^2+\tau^2\mathcal{V}_{\mathcal{F},1}+\mathbb{E}\left[\left\|\gamma_t D_{\lambda,t}\right\|^2\mid\mathfrak{S}_t\right]\right)
\end{aligned}\tag{6.122}$$

The proof of the second inequality follows from the same arguments as in the deterministic drift analysis. We omit the details for brevity. □

**Lemma 6.23.** (Stochastic control of the dual error) *Assume that the lower step size satisfies* $\tau\leqslant 1/L^{w,w}_{\mathcal{F},1}$ *and that the dual step size satisfies* $\rho\leqslant\min\left(1/L^{w,w}_{\mathcal{F},1},\mu_{\mathcal{F}}/4\mathcal{V}^{w,w}_{\mathcal{F},2}\right)$. *For any* $t\geqslant 0$, *we have*

$$\begin{aligned}\mathbb{E}\Big[\|\nu_{t+1}-\nu^*(\lambda_{t+1})\|^2\Big] \leqslant{}& (1-\rho\mu_{\mathcal{F}}/2)\|\nu_t-\nu^*(\lambda_t)\|^2\\ &+28\rho^{-1}\mu_{\mathcal{F}}^{-1}L_{\nu,*}\Big(2+\tau^2\Big(L_{\mathcal{F},1}^{w,w}\Big)^2\Big)\|w_t-w_t^*\|^2\\ &+27\rho^{-1}\mu_{\mathcal{F}}^{-1}L_{\nu,*}\Big(1+3\tau^{-1}\mu_{\mathcal{F}}^{-3}M_{\mathcal{C}}^2\Big)\mathbb{E}\Big[\big\|\gamma_t D_{\lambda,t}\big\|^2 \mid \mathfrak{S}_t\Big]\\ &+27\tau^2\rho^{-1}\mu_{\mathcal{F}}^{-1}L_{\nu,*}\frac{\kappa_{\mathcal{F},w}}{\kappa_{\mathcal{F},w}-1}\mathcal{V}_{\mathcal{F},1}\\ &+4\rho^2\Big(\mathcal{V}_{\mathcal{U},1}^{w}+\mathcal{V}_{\mathcal{F},2}^{w,w}\mathcal{C}_{\nu,*}\Big)\end{aligned} \tag{6.123}$$

*where* $L_{\nu,*}$ *is the constant defined in Lemma 6.22.*

*Proof.* We follow similar steps to the deterministic control of the dual error. Using Lemma 6.20 and Young's inequality, we have for any $\alpha,\beta,\delta>0$,

$$\begin{aligned}\mathbb{E}\Big[\|\nu_{t+1}-\nu^*(\lambda_{t+1})\|^2 \mid \mathfrak{S}_t\Big] \leqslant{}& (1+\alpha)(1+\beta)(1+\delta)(1-\rho\mu_{\mathcal{F}})\|\nu_t-\nu^*(\lambda_t)\|^2\\ &+(1+\alpha)(1+\beta)\big(1+\delta^{-1}\big)(1-\rho\mu_{\mathcal{F}})\|\nu^*(\lambda_t)-\nu^*(\lambda_t,w_t)\|^2\\ &+(1+\alpha)(1+\beta)\rho^2\Big(\mathcal{V}_{\mathcal{F},2}^{w,w}\|\nu_t\|^2+\mathcal{V}_{\mathcal{U},1}^{w}\Big)\\ &+(1+\alpha)\big(1+\beta^{-1}\big)\mathbb{E}\Big[\|\nu^*(\lambda_t,w_t)-\nu^*(\lambda_{t+1},w_{t+1})\|^2 \mid \mathfrak{S}_t\Big]\\ &+\big(1+\alpha^{-1}\big)\mathbb{E}\Big[\|\nu^*(\lambda_{t+1},w_{t+1})-\nu^*(\lambda_{t+1})\|^2 \mid \mathfrak{S}_t\Big].\end{aligned} \tag{6.124}$$

Using Lemma 6.22, we deduce that, for any $\rho\leqslant 1/L_{\mathcal{F},1}^{w,w}$,

$$\begin{aligned}\mathbb{E}\Big[\|\nu_{t+1}-\nu^*(\lambda_{t+1})\|^2 \mid \mathfrak{S}_t\Big] \leqslant{}& (1+\alpha)(1+\beta)(1+\delta)(1-\rho\mu_{\mathcal{F}})\|\nu_t-\nu^*(\lambda_t)\|^2\\ &+(1+\alpha)(1+\beta)\big(1+\delta^{-1}\big)(1-\rho\mu_{\mathcal{F}})L_{\nu,*}\|w_t-w_t^*\|^2\\ &+(1+\alpha)(1+\beta)\rho^2\Big(\mathcal{V}_{\mathcal{F},2}^{w,w}\|\nu_t\|^2+\mathcal{V}_{\mathcal{U},1}^{w}\Big)\\ &+(1+\alpha)\big(1+\beta^{-1}\big)L_{\nu,*}\Big(\tau^2\Big(L_{\mathcal{F},1}^{w,w}\Big)^2\|w_t-w_t^*\|^2\\ &\quad+\tau^2\mathcal{V}_{\mathcal{F},1}+\mathbb{E}\Big[\big\|\gamma_t D_{\lambda,t}\big\|^2 \mid \mathfrak{S}_t\Big]\Big)\\ &+\big(1+\alpha^{-1}\big)L_{\nu,*}\mathbb{E}\Big[\big\|w_{t+1}-w_{t+1}^*\big\|^2 \mid \mathfrak{S}_t\Big].\end{aligned} \tag{6.125}$$

Using once more Lemma 6.20, we obtain for any $\tau\leqslant 1/L_{\mathcal{F},1}^{w,w}$,

$$\begin{aligned}
\mathbb{E}\Big[\|v_{t+1}-v^*(\lambda_{t+1})\|^2 \mid \mathfrak{S}_t\Big] \leqslant\ & (1+\alpha)(1+\beta)(1+\delta)(1-\rho\mu_{\mathcal{F}})\|v_t-v^*(\lambda_t)\|^2 \\
& +(1+\alpha)(1+\beta)\big(1+\delta^{-1}\big)(1-\rho\mu_{\mathcal{F}})L_{v,*}\|w_t-w_t^*\|^2 \\
& +(1+\alpha)(1+\beta)\rho^2\Big(\mathcal{V}_{\mathcal{F},2}^{w,w}\|v_t\|^2+\mathcal{V}_{\mathcal{U},1}^{w}\Big) \\
& +(1+\alpha)\big(1+\beta^{-1}\big)L_{v,*}\Big(\tau^2\Big(L_{\mathcal{F},1}^{w,w}\Big)^2\|w_t-w_t^*\|^2 \\
& \quad +\tau^2\mathcal{V}_{\mathcal{F},1}+\mathbb{E}\Big[\big\|\gamma_t D_{\lambda,t}\big\|^2 \mid \mathfrak{S}_t\Big]\Big) \\
& +\big(1+\alpha^{-1}\big)L_{v,*}\Big[(1-\tau\mu_{\mathcal{F}}/2)\|w_t-w_t^*\|^2 \\
& \quad +3\tau^{-1}\mu_{\mathcal{F}}^{-3}M_{\mathcal{C}}^2\mathbb{E}\Big[\big\|\gamma_t D_{\lambda,t}\big\|^2 \mid \mathfrak{S}_t\Big] \\
& \quad +\tau^2\frac{\kappa_{\mathcal{F},w}}{\kappa_{\mathcal{F},w}-1}\mathcal{V}_{\mathcal{F},1}\Big].
\end{aligned} \tag{6.126}$$

Setting again $\alpha=\beta=\delta=\rho\mu_{\mathcal{F}}/24$ and using the bounds derived in the deterministic control of the dual error, we obtain:

$$\begin{aligned}
\mathbb{E}\Big[\|v_{t+1}-v^*(\lambda_{t+1})\|^2 \mid \mathfrak{S}_t\Big] \leqslant\ & (1-\rho\mu_{\mathcal{F}}/2)\|v_t-v^*(\lambda_t)\|^2 \\
& +28\rho^{-1}\mu_{\mathcal{F}}^{-1}(1-\rho\mu_{\mathcal{F}})L_{v,*}\|w_t-w_t^*\|^2 \\
& +4\rho^2\Big(\mathcal{V}_{\mathcal{F},2}^{w,w}\|v_t\|^2+\mathcal{V}_{\mathcal{U},1}^{w}\Big) \\
& +27\rho^{-1}\mu_{\mathcal{F}}^{-1}L_{v,*}\Big(\tau^2\Big(L_{\mathcal{F},1}^{w,w}\Big)^2\|w_t-w_t^*\|^2 \\
& \quad +\tau^2\mathcal{V}_{\mathcal{F},1}+\mathbb{E}\Big[\big\|\gamma_t D_{\lambda,t}\big\|^2 \mid \mathfrak{S}_t\Big]\Big) \\
& +25\rho^{-1}\mu_{\mathcal{F}}^{-1}L_{v,*}\Big[(1-\tau\mu_{\mathcal{F}}/2)\|w_t-w_t^*\|^2 \\
& \quad +3\tau^{-1}\mu_{\mathcal{F}}^{-3}M_{\mathcal{C}}^2\mathbb{E}\Big[\big\|\gamma_t D_{\lambda,t}\big\|^2 \mid \mathfrak{S}_t\Big] \\
& \quad +\tau^2\frac{\kappa_{\mathcal{F},w}}{\kappa_{\mathcal{F},w}-1}\mathcal{V}_{\mathcal{F},1}\Big] \\
\leqslant\ & (1-\rho\mu_{\mathcal{F}}/2)\|v_t-v^*(\lambda_t)\|^2 \\
& +28\rho^{-1}\mu_{\mathcal{F}}^{-1}L_{v,*}\Big(2+\tau^2\Big(L_{\mathcal{F},1}^{w,w}\Big)^2\Big)\|w_t-w_t^*\|^2 \\
& +27\rho^{-1}\mu_{\mathcal{F}}^{-1}L_{v,*}\Big(1+3\tau^{-1}\mu_{\mathcal{F}}^{-3}M_{\mathcal{C}}^2\Big)\mathbb{E}\Big[\big\|\gamma_t D_{\lambda,t}\big\|^2\Big] \\
& +4\rho^2\mathcal{V}_{\mathcal{F},2}^{w,w}\|v_t\|^2+4\rho^2\mathcal{V}_{\mathcal{U},1}^{w}+27\rho^{-1}\mu_{\mathcal{F}}^{-1}L_{v,*}\tau^2\frac{2\kappa_{\mathcal{F},w}}{\kappa_{\mathcal{F},w}-1}\mathcal{V}_{\mathcal{F},1}
\end{aligned} \tag{6.127}$$

where we additionally used that $(1+\alpha)(1+\beta)=(1+\alpha)^2\leqslant 4$. The desired result follows from the upper bound in Lemma 6.21:

$$\begin{aligned}\mathbb{E}\left[\|\nu_{t+1}-\nu^*(\lambda_{t+1})\|^2\right] \leqslant (1-\rho\mu_{\mathcal{F}}/2)\|\nu_t-\nu^*(\lambda_t)\|^2 \\ +28\rho^{-1}\mu_{\mathcal{F}}^{-1}L_{v,*}\left(2+\tau^2\left(L_{\mathcal{F},1}^{w,w}\right)^2\right)\|w_t-w_t^*\|^2 \\ +27\rho^{-1}\mu_{\mathcal{F}}^{-1}L_{v,*}\left(1+3\tau^{-1}\mu_{\mathcal{F}}^{-3}M_{\mathcal{C}}^2\right)\mathbb{E}\left[\left\|\gamma_t D_{\lambda,t}\right\|^2\right] \\ +27\tau^2\rho^{-1}\mu_{\mathcal{F}}^{-1}L_{v,*}\frac{\kappa_{\mathcal{F},w}}{\kappa_{\mathcal{F},w}-1}\mathcal{V}_{\mathcal{F},1} \\ +4\rho^2\left(\mathcal{V}_{\mathcal{U},1}^{w}+\mathcal{V}_{\mathcal{F},2}^{w,w}\mathcal{C}_{v,*}\right)\end{aligned} \tag{6.128}$$

□

### Stochastic control of the error on the implicit gradient estimate

We now extend Lemma 6.18 to the stochastic setting.

**Lemma 6.24.** (Stochastic error bound on the implicit gradient) *For any* $t \geqslant 0$*, we have*

$$\begin{aligned}\mathbb{E}\left[\left\|\nabla\phi(\lambda_t)-D_{\lambda,t}^{\text{in}}\right\|^2\right] \leqslant \quad & 2\left(L_{\mathcal{U},1}\right)^2\mathbb{E}\left[\|w_t-w_t^*\|^2\right]+4M_{\mathcal{C}}^2\mathbb{E}\left[\|\nu_t-\nu^*(\lambda_t)\|^2\right] \\ +4\left(L_{\mathcal{F},2}^{w,\lambda}\right)^2&\mathbb{E}\left[\|w_t-w_t^*\|^2\cdot\|\nu_t\|^2\right] \\ +\mathcal{V}_{\mathcal{U},1}^{\lambda}&+\mathcal{V}_{\mathcal{F},2}^{w,\lambda}\mathcal{C}_{v,*}.\end{aligned} \tag{6.129}$$

*and* $\mathbb{E}\left[\left\|D_{\lambda,t}-\bar{D}_{\lambda,t}\right\|^2\right]$ *satisfies the same upper bound.*

*Proof.* First note that, by Assumption 6.7, we have $\mathbb{E}\left[D_{\lambda,t}^{\text{in}} \mid \mathfrak{S}_{t,\nu}\right] = \nabla_\lambda\mathcal{U}(w_t,\lambda_t) + \nabla^2_{w,\lambda}\mathcal{F}(w_t,\lambda_t)^\top\nu_t$.

Hence, for any $t \geqslant 0$,

$$\begin{aligned}&\mathbb{E}\left[\left\|\nabla\phi(\lambda_t)-\mathbb{E}\left[D_{\lambda,t}^{\text{in}} \mid \mathfrak{S}_{t,\nu}\right]\right\|^2 \mid \mathfrak{S}_{t,\nu}\right] \\ &\leqslant 2\|\nabla_\lambda\mathcal{U}(w_t^*,\lambda_t)-\nabla_\lambda\mathcal{U}(w_t,\lambda_t)\|^2 \\ &\quad+2\left\|\nabla^2_{w,\lambda}\mathcal{F}(w_t^*,\lambda_t)^\top\nu^*(\lambda_t)-\nabla^2_{w,\lambda}\mathcal{F}(w_t,\lambda_t)^\top\nu_t\right\|^2 \\ &\leqslant 2\left(L_{\mathcal{U},1}\right)^2\|w_t-w_t^*\|^2+4M_{\mathcal{C}}^2\|\nu_t-\nu^*(\lambda_t)\|^2 \\ &\quad+4\left(L_{\mathcal{F},2}^{w,\lambda}\right)^2\|w_t-w_t^*\|^2\cdot\|\nu_t\|^2.\end{aligned} \tag{6.130}$$

Furthermore, we have by Assumption 6.7

$$\begin{aligned}\mathbb{E}\left[\left\|\mathbb{E}\left[D^{in}_{\lambda,t} \mid \mathfrak{S}_{t,v}\right] - D^{in}_{\lambda,t}\right\|^2 \mid \mathfrak{S}_{t,v}\right] &\leqslant \mathbb{E}\left[\left\|\nabla_\lambda \mathcal{U}(w_t,\lambda_t) - \tilde{\nabla}_\lambda \mathcal{U}(w_t,\lambda_t)\right\|^2 \mid \mathfrak{S}_{t,v}\right] \\ &\quad + \mathbb{E}\left[\left\|\nabla^2_{w,\lambda}\mathcal{F}(w_t,\lambda_t) - \tilde{\nabla}^2_{w,\lambda}\mathcal{F}(w_t,\lambda_t)\right\|^2 \cdot \|v_t\|^2 \mid \mathfrak{S}_{t,v}\right] \\ &\leqslant \mathcal{V}^\lambda_{\mathcal{U},1} + \mathcal{V}^{w,\lambda}_{\mathcal{F},2}\|v_t\|^2.\end{aligned} \tag{6.131}$$

Hence, by standard properties of the conditional expectation, we obtain

$$\begin{aligned}&\mathbb{E}\left[\left\|\nabla\phi(\lambda_t) - D^{in}_{\lambda,t}\right\|^2\right] \\ &\leqslant 2\big(L_{\mathcal{U},1}\big)^2\mathbb{E}\left[\|w_t - w^*_t\|^2\right] + 4M^2_{\mathcal{C}}\mathbb{E}\left[\|v_t - v^*(\lambda_t)\|^2\right] + 4\Big(L^{w,\lambda}_{\mathcal{F},2}\Big)^2\mathbb{E}\left[\|w_t - w^*_t\|^2 \cdot \|v_t\|^2\right] \\ &+\mathcal{V}^\lambda_{\mathcal{U},1} + \mathcal{V}^{w,\lambda}_{\mathcal{F},2}\mathcal{C}_{v,*}.\end{aligned} \tag{6.132}$$

where last line follows again from Lemma 6.21.

The same upper bound is satisfied by $\mathbb{E}\left[\left\|D_{\lambda,t} - \bar{D}_{\lambda,t}\right\|^2\right]$ thanks to the contraction properties of the projection operator on $\mathcal{C}$. □

Compared with Lemma 6.18, we have now an additional bound on $\mathbb{E}\left[\|w_t - w^*_t\|^2 \cdot \|v_t\|^2\right]$.

The following lemma allows us to control it recursively.

**Lemma 6.25.** (Stochastic control of the joint dynamic) *For any* $t \geqslant 0$, *we have*

$$\begin{aligned}&\mathbb{E}\left[\left\|w_{t+1} - w^*_{t+1}\right\|^2 \cdot \|v_{t+1}\|^2\right] \\ &\leqslant \left(1 - \frac{\tau\mu_{\mathcal{F}}}{2}\right)\left(1 - \rho\frac{\mu_{\mathcal{F}}}{8}\right)\mathbb{E}\left[\|w_t - w^*_t\|^2 \cdot \|v_t\|^2\right] \\ &+\left(1 - \rho\frac{\mu_{\mathcal{F}}}{8}\right)\cdot 3\tau^{-1}\mu_{\mathcal{F}}^{-3}M^2_{\mathcal{C}}\mathbb{E}\left[\mathbb{E}\left[\left\|\gamma_t D_{\lambda,t}\right\|^2 \mid \mathfrak{S}_t\right]\cdot\|v_t\|^2\right] \\ &+\left(1 - \rho\frac{\mu_{\mathcal{F}}}{8}\right)\tau^2\frac{\kappa_{\mathcal{F},w}}{\kappa_{\mathcal{F},w} - 1}\mathcal{V}_{\mathcal{F},1}\mathbb{E}\left[\|v_t\|^2\right] \\ &+\left(1 - \frac{\tau\mu_{\mathcal{F}}}{2}\right)\left(\frac{15}{\rho\mu_{\mathcal{F}}}\left(\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}}\right)^2 + 2\rho^2\mathcal{V}^w_{\mathcal{U},1}\right)\mathbb{E}\left[\|w_t - w^*_t\|^2\right] \\ &+3\tau^{-1}\mu_{\mathcal{F}}^{-3}M^2_{\mathcal{C}}\left(\frac{15}{\rho\mu_{\mathcal{F}}}\left(\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}}\right)^2 + 2\rho^2\mathcal{V}^w_{\mathcal{U},1}\right)\mathbb{E}\left[\mathbb{E}\left[\left\|\gamma_t D_{\lambda,t}\right\|^2 \mid \mathfrak{S}_t\right]\right] \\ &+\tau^2\left(\frac{15}{\rho\mu_{\mathcal{F}}}\left(\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}}\right)^2 + 2\rho^2\mathcal{V}^w_{\mathcal{U},1}\right)\frac{\kappa_{\mathcal{F},w}}{\kappa_{\mathcal{F},w} - 1}\mathcal{V}_{\mathcal{F},1}\end{aligned} \tag{6.133}$$

*Proof.* First observe that, in the first upper bound given in Lemma 6.21, one can condition on $\mathfrak{S}_{t,w}$ instead of $\mathfrak{S}_t$ without affecting the bound, so that,

$$\mathbb{E}\Big[\|w_{t+1}-w^*_{t+1}\|^2\cdot\|v_{t+1}\|^2\Big]=\mathbb{E}\Big[\|w_{t+1}-w^*_{t+1}\|^2\cdot\mathbb{E}\Big[\|v_{t+1}\|^2\mid\mathfrak{S}_{t,w}\Big]\Big]$$
$$\leqslant\mathbb{E}\Bigg[\|w_{t+1}-w^*_{t+1}\|^2\bigg(\Big(1-\rho\frac{\mu_{\mathcal{F}}}{8}\Big)\|v_t\|^2+\frac{15}{\rho\mu_{\mathcal{F}}}\Big(\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}}\Big)^2+2\rho^2\mathcal{V}^w_{\mathcal{U},1}\bigg)\Bigg]. \tag{6.134}$$

Thus, conditioning now on $\mathfrak{S}_t$ and using Lemma 6.20, we obtain

$$\begin{aligned}
&\mathbb{E}\Big[\|w_{t+1}-w^*_{t+1}\|^2\cdot\|v_{t+1}\|^2\Big]\\
&\quad\leqslant\mathbb{E}\Bigg[\bigg(\Big(1-\frac{\tau\mu_{\mathcal{F}}}{2}\Big)\|w_t-w^*_t\|^2+3\tau^{-1}\mu_{\mathcal{F}}^{-3}M_{\mathcal{C}}^2\mathbb{E}\Big[\|\gamma_t D_{\lambda,t}\|^2\mid\mathfrak{S}_t\Big]+\tau^2\frac{\kappa_{\mathcal{F},w}}{\kappa_{\mathcal{F},w}-1}\mathcal{V}_{\mathcal{F},1}\bigg)\\
&\qquad\cdot\bigg(\Big(1-\rho\frac{\mu_{\mathcal{F}}}{8}\Big)\|v_t\|^2+\frac{15}{\rho\mu_{\mathcal{F}}}\Big(\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}}\Big)^2+2\rho^2\mathcal{V}^w_{\mathcal{U},1}\bigg)\Bigg]\\
&\quad\leqslant\Big(1-\frac{\tau\mu_{\mathcal{F}}}{2}\Big)\Big(1-\rho\frac{\mu_{\mathcal{F}}}{8}\Big)\mathbb{E}\Big[\|w_t-w^*_t\|^2\cdot\|v_t\|^2\Big]\\
&+\Big(1-\rho\frac{\mu_{\mathcal{F}}}{8}\Big)\cdot3\tau^{-1}\mu_{\mathcal{F}}^{-3}M_{\mathcal{C}}^2\mathbb{E}\Big[\mathbb{E}\Big[\|\gamma_t D_{\lambda,t}\|^2\mid\mathfrak{S}_t\Big]\cdot\|v_t\|^2\Big]\\
&+\Big(1-\rho\frac{\mu_{\mathcal{F}}}{8}\Big)\tau^2\frac{\kappa_{\mathcal{F},w}}{\kappa_{\mathcal{F},w}-1}\mathcal{V}_{\mathcal{F},1}\mathbb{E}\Big[\|v_t\|^2\Big]\\
&+\Big(1-\frac{\tau\mu_{\mathcal{F}}}{2}\Big)\bigg(\frac{15}{\rho\mu_{\mathcal{F}}}\Big(\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}}\Big)^2+2\rho^2\mathcal{V}^w_{\mathcal{U},1}\bigg)\mathbb{E}\Big[\|w_t-w^*_t\|^2\Big]\\
&+3\tau^{-1}\mu_{\mathcal{F}}^{-3}M_{\mathcal{C}}^2\bigg(\frac{15}{\rho\mu_{\mathcal{F}}}\Big(\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}}\Big)^2+2\rho^2\mathcal{V}^w_{\mathcal{U},1}\bigg)\mathbb{E}\Big[\mathbb{E}\Big[\|\gamma_t D_{\lambda,t}\|^2\mid\mathfrak{S}_t\Big]\Big]\\
&+\tau^2\bigg(\frac{15}{\rho\mu_{\mathcal{F}}}\Big(\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}}\Big)^2+2\rho^2\mathcal{V}^w_{\mathcal{U},1}\bigg)\frac{\kappa_{\mathcal{F},w}}{\kappa_{\mathcal{F},w}-1}\mathcal{V}_{\mathcal{F},1}
\end{aligned} \tag{6.135}$$

□

### Proof of the stochastic convergence theorem

We can now present the proof of Theorem 6.8.

*Proof.* For the stochastic setting, we consider the Lyapunov function

$$\psi_t\triangleq\mathbb{E}[\phi(\lambda_t)]+\alpha\mathbb{E}\Big[\|w_t-w^*_t\|^2\Big]+\beta\mathbb{E}\Big[\|v_t-v^*(\lambda_t)\|^2\Big]+\delta\mathbb{E}\Big[\|w_t-w^*_t\|^2\cdot\|v_t\|^2\Big] \tag{6.136}$$

where $\alpha,\beta,\delta$ are nonnegative parameters to be fixed subsequently in the analysis.

By Lemma 6.27, we have for all $t\geqslant0$,

$$\begin{aligned}
\psi_{t+1}\leqslant\psi_t&-\mathbb{E}\Big[\frac{\gamma_t}{8}\|D_{\lambda,t}\|^2\Big]-\mathbb{E}\Big[\frac{\gamma_t}{8}\|\bar{D}_{\lambda,t}\|^2\Big]-\frac{\alpha\tau\mu_{\mathcal{F}}}{4}\mathbb{E}\Big[\|w_t-w^*_t\|^2\Big]+\mathbb{E}\big[\mathcal{C}_{\psi,t}\big]\\
&-\mathbb{E}\Big[\mathcal{C}_{D,t}\|D_{\lambda,t}\|^2+\mathcal{C}_{w,t}\|w_t-w^*_t\|^2+\mathcal{C}_{v,t}\|v_t-v^*(\lambda_t)\|^2+\mathcal{C}_{w,v,t}\|w_t-w^*_t\|^2\cdot\|v_t\|^2\Big]
\end{aligned} \tag{6.137}$$

where the five quantities $\mathcal{C}_{D,t},\mathcal{C}_{w,t},\mathcal{C}_{v,t},\mathcal{C}_{w,v,t}$ and $\mathcal{C}_{\psi,t}$ are specified in Lemma 6.27.

To ensure the non-negativity of the first four ones, we set $\alpha, \beta, \delta$ as specified in Lemma 6.28, together with the parameterization $\tau = \frac{1}{L^{w,w}_{\mathcal{F},1}}$, $\rho = \min\left(\frac{1}{L^{w,w}_{\mathcal{F},1}}, \frac{\mu_{\mathcal{F}}}{4\mathcal{V}^{w,w}_{\mathcal{F},2}}\right)$ and $\gamma = \min\left(\bar{\gamma}, L_{\phi}^{-1}\right)$ where

$$
\begin{aligned}
(\bar{\gamma})^{-1} \triangleq\ & 2L_{\phi} + 72M_{\mathcal{C}}^{2}\kappa_{\mathcal{F},w}^{2}\left(L_{\mathcal{U},1}\right)^{2}L_{\phi}^{-1}\mu_{\mathcal{F}}^{-2} \\
& + 32256L_{v,*}M_{\mathcal{C}}^{4}\kappa_{\mathcal{F},w}^{2}L_{\phi}^{-1}\mu_{\mathcal{F}}^{-2}\left(\kappa_{\mathcal{F},w} + 4\mu_{\mathcal{F}}^{-2}\mathcal{V}^{w}_{\mathcal{U},1}\right)^{2} \\
& + 288\kappa_{\mathcal{F},w}^{2}M_{\mathcal{C}}^{2}\left(L^{w,w}_{\mathcal{F},2}\right)^{2}L_{\phi}^{-1}\mu_{\mathcal{F}}^{-2}\left(15\left(\kappa_{\mathcal{F},w} + 4\mu_{\mathcal{F}}^{-2}\mathcal{V}^{w}_{\mathcal{U},1}\right)\left(\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}}\right)^{2} + 2\frac{\mathcal{V}^{w}_{\mathcal{U},1}}{\left(L^{w,w}_{\mathcal{F},1}\right)^{2}}\right) \\
& + 648L_{v,*}M_{\mathcal{C}}^{2}L_{\phi}^{-1}\left(1 + 3\kappa_{\mathcal{F},w}\mu_{\mathcal{F}}^{-2}M_{\mathcal{C}}^{2}\right)\left(\kappa_{\mathcal{F},w} + 4\mu_{\mathcal{F}}^{-2}\mathcal{V}^{w}_{\mathcal{U},1}\right)^{2} \\
& + 72\kappa_{\mathcal{F},w}^{2}\left(L^{w,w}_{\mathcal{F},2}\right)^{2}M_{\mathcal{C}}^{2}L_{\phi}^{-1}\mu_{\mathcal{F}}^{-2}\left(15\left(\kappa_{\mathcal{F},w} + 4\mu_{\mathcal{F}}^{-2}\mathcal{V}^{w}_{\mathcal{U},1}\right)\left(\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}}\right)^{2} + 2\frac{\mathcal{V}^{w}_{\mathcal{U},1}}{\left(L^{w,w}_{\mathcal{F},1}\right)^{2}}\right).
\end{aligned}
\tag{6.138}
$$

Under this parameterization, the remaining term $\mathcal{C}_{\psi,t}$ can be bounded above by a constant $\mathcal{C}_{\psi}$ we specify in Lemma 6.29.

This allows us to unroll (6.137) to get

$$
\frac{1}{T}\sum_{t=0}^{T-1}\left[\mathbb{E}\left[\frac{\gamma_t}{8}\left(\left\|\bar{D}_{\lambda,t}\right\|^{2} + \left\|D_{\lambda,t}\right\|^{2}\right)\right] + \frac{\alpha\tau\mu_{\mathcal{F}}}{4}\mathbb{E}\left[\left\|w_t - w_t^{*}\right\|^{2}\right]\right] \leqslant \frac{1}{T}\left(\psi_0 - \inf_{\mathcal{C}}\phi\right) + \mathcal{C}_{\psi}. \tag{6.139}
$$

This implies

$$
\begin{aligned}
\frac{1}{T}\sum_{t=0}^{T-1}\left[\mathbb{E}\left[\mathbb{1}_{\|D_{\lambda,t}\|\leqslant C_{\gamma}}\left(\left\|\bar{D}_{\lambda,t}\right\|^{2} + \left\|D_{\lambda,t}\right\|^{2}\right)\right] + \frac{\alpha\tau\mu_{\mathcal{F}}}{4\gamma}\mathbb{E}\left[\left\|w_t - w_t^{*}\right\|^{2}\right]\right] &\leqslant \frac{1}{\gamma T}\left(\psi_0 - \inf_{\mathcal{C}}\phi\right) + \frac{\mathcal{C}_{\psi}}{\gamma} \\
\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\frac{C_{\gamma}}{\left\|D_{\lambda,t}\right\|}\mathbb{1}_{\|D_{\lambda,t}\|>C_{\gamma}}\left(\left\|\bar{D}_{\lambda,t}\right\|^{2} + \left\|D_{\lambda,t}\right\|^{2}\right)\right] &\leqslant \frac{1}{\gamma T}\left(\psi_0 - \inf_{\mathcal{C}}\phi\right) + \frac{\mathcal{C}_{\psi}}{\gamma}.
\end{aligned}
\tag{6.140}
$$

Using the standard inequality $a^2 \geqslant 2\epsilon a - \epsilon^2$ for $a, \epsilon \geqslant 0$, we derive from (6.140) that

$$
\frac{1}{T}\sum_{t=0}^{T-1}\left[\mathbb{E}\left[\mathbb{1}_{\|D_{\lambda,t}\|\leqslant C_{\gamma}}\left\|\bar{D}_{\lambda,t}\right\|\right] + \frac{1}{2}\left(\frac{\alpha\tau\mu_{\mathcal{F}}}{\gamma}\right)^{\frac{1}{2}}\mathbb{E}[\|w_t - w_t^{*}\|]\right] \leqslant \left(\frac{1}{\gamma T}\left(\psi_0 - \inf_{\mathcal{C}}\phi\right) + \frac{\mathcal{C}_{\psi}}{\gamma}\right)^{\frac{1}{2}}, \tag{6.141}
$$

where we set $\epsilon = \left(\frac{1}{T}(\psi_0 - \inf_{\mathcal{C}}\phi) + \mathcal{C}_{\psi}\right)^{\frac{1}{2}}$.

Furthermore, noting that $\frac{1}{\|D_{\lambda,t}\|}\left(\left\|\bar{D}_{\lambda,t}\right\|^{2} + \left\|D_{\lambda,t}\right\|^{2}\right) \geqslant \frac{2}{\|D_{\lambda,t}\|}\left(\left\|\bar{D}_{\lambda,t}\right\| \cdot \left\|D_{\lambda,t}\right\|\right) = 2\left\|\bar{D}_{\lambda,t}\right\|$, we derive from (6.140) that

$$
\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\mathbb{1}_{\|D_{\lambda,t}\|>C_{\gamma}}\left\|\bar{D}_{\lambda,t}\right\|\right] \leqslant \frac{1}{C_{\gamma}}\left(\frac{1}{\gamma T}\left(\psi_0 - \inf_{\mathcal{C}}\phi\right) + \frac{\mathcal{C}_{\psi}}{\gamma}\right). \tag{6.142}
$$

Combining these two inequalities allows us to conclude that:

$$\begin{aligned}\frac{1}{T}\sum_{t=0}^{T-1}\left[\mathbb{E}\left[\|\bar{D}_{\lambda,t}\|\right]+\frac{1}{2}\left(\frac{\alpha\tau\mu_{\mathcal{F}}}{\gamma}\right)^{\frac{1}{2}}\mathbb{E}[\|w_t-w_t^*\|]\right]\\ \leqslant\left(\frac{1}{\gamma T}\Big(\psi_0-\inf_{\mathcal{C}}\phi\Big)+\frac{\mathcal{C}_\psi}{\gamma}\right)^{\frac{1}{2}}\\ +\frac{1}{C_\gamma}\left(\frac{1}{\gamma T}\Big(\psi_0-\inf_{\mathcal{C}}\phi\Big)+\frac{\mathcal{C}_\psi}{\gamma}\right),\end{aligned}\tag{6.143}$$

which writes as (6.26) for $\left[\tilde{\alpha},\tilde{\beta},\tilde{\delta}\right]=\frac{1}{4\gamma\kappa_{\mathcal{F}}}[\alpha,\beta,\delta]$, i.e.

$$\begin{aligned}\tilde{\alpha}=\Bigg(&\frac{3}{2}(L_{\mathcal{U},1})^2+672\Big(\kappa_{\mathcal{F},w}+4\mu_{\mathcal{F}}^{-2}\nu_{\mathcal{U},1}^w\Big)^2L_{\nu,*}M_{\mathcal{C}}^2\\ &+6\Big(L_{\mathcal{F},2}^{w,w}\Big)^2\left(15\Big(\kappa_{\mathcal{F},w}+4\mu_{\mathcal{F}}^{-2}\nu_{\mathcal{U},1}^w\Big)\left(\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}}\right)^2+2\frac{\nu_{\mathcal{U},1}^w}{\Big(L_{\mathcal{F},1}^{w,w}\Big)^2}\right)\Bigg)\\ \tilde{\beta}=\frac{3}{2}&\Big(1+4\Big(L_{\mathcal{F},1}^{w,w}\Big)^{-1}\mu_{\mathcal{F}}^{-1}\nu_{\mathcal{U},1}^w\Big)M_{\mathcal{C}}^2,\quad\tilde{\delta}=\frac{3}{2}\Big(L_{\mathcal{F},2}^{w,w}\Big)^2.\end{aligned}\tag{6.144}$$

and $M=[M_1,M_2,M_3,M_4,M_5]^\top$ with the entries given by $M_1=\frac{3}{4}$, $M_2=24M_{\mathcal{C}}^2\Big(L_{\mathcal{F},1}^{w,w}\Big)^{-1}\mu_{\mathcal{F}}^{-1}$, $M_4=\frac{3}{4}\mathcal{C}_{\nu,*}$, $M_5=24\mathcal{C}_{\nu,*}M_{\mathcal{C}}^2\Big(L_{\mathcal{F},1}^{w,w}\Big)^{-1}\mu_{\mathcal{F}}^{-1}$, and

$$\begin{aligned}M_3=\Bigg[&6L_{\mathcal{U},1}^2\frac{1}{\kappa_{\mathcal{F},w}-1}\mu_{\mathcal{F}}^{-2}+2688L_{\nu,*}M_{\mathcal{C}}^2\frac{\Big(\kappa_{\mathcal{F},w}+4\mu_{\mathcal{F}}^{-2}\nu_{\mathcal{U},1}^w\Big)^2}{(\kappa_{\mathcal{F},w}-1)\mu_{\mathcal{F}}^2}\\ &+24\frac{\kappa_{\mathcal{F},w}^2}{\kappa_{\mathcal{F},w}-1}\left(15\Big(\kappa_{\mathcal{F},w}+4\mu_{\mathcal{F}}^{-2}\nu_{\mathcal{U},1}^w\Big)\left(\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}}\right)^2+2\frac{\nu_{\mathcal{U},1}^w}{\Big(L_{\mathcal{F},1}^{w,w}\Big)^2}\right)\\ &+324L_{\nu,*}M_{\mathcal{C}}^2\frac{\Big(\kappa_{\mathcal{F},w}+4\mu_{\mathcal{F}}^{-2}\nu_{\mathcal{U},1}^w\Big)^2}{(\kappa_{\mathcal{F},w}-1)\mu_{\mathcal{F}}}\Big(L_{\mathcal{F},1}^{w,w}\Big)^{-1}+6\mathcal{C}_{\nu,*}\Big(L_{\mathcal{F},2}^{w,w}\Big)^2\frac{1}{(\kappa_{\mathcal{F},w}-1)\mu_{\mathcal{F}}^2}\\ &+6\Big(L_{\mathcal{F},2}^{w,w}\Big)^2\frac{1}{(\kappa_{\mathcal{F},w}-1)\mu_{\mathcal{F}}^2}\left(15\Big(\kappa_{\mathcal{F},w}+4\mu_{\mathcal{F}}^{-2}\nu_{\mathcal{U},1}^w\Big)\left(\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}}\right)^2+2\frac{\nu_{\mathcal{U},1}^w}{\Big(L_{\mathcal{F},1}^{w,w}\Big)^2}\right)\Bigg],\end{aligned}\tag{6.145}$$

□

### 6.B.3 Derivation of the Lyapunov bounds

We provide the derivation steps for the Lyapunov decrease arguments presented in Section 6.B.

Parameter policy for the deterministic setting

**Lemma 6.26.** *Assume* $\tau = \rho = \frac{1}{L^{w,w}_{\mathcal{F},1}}$ *and* $\gamma \leqslant \min\Big(\frac{1}{\kappa_{\mathcal{F},w}}, \bar{\gamma}\Big)$ *where*

$$\begin{aligned}\bar{\gamma}^{-1} = 2L_\phi + 48\kappa_{\mathcal{F},w}\left(\frac{M_{\mathcal{C}}}{\mu_{\mathcal{F}}}\right)^2\left(L_{\mathcal{U},1} + L^{w,\lambda}_{\mathcal{F},2}\left(4\kappa_{\mathcal{F},w}\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}} + \epsilon_{v,0}\right)\right)^2 \\ +10272\kappa^3_{\mathcal{F},w}\frac{M^4_{\mathcal{C}}}{\mu^2_{\mathcal{F}}}M^2_{v^*} + 300\kappa_{\mathcal{F},w}M^2_{\mathcal{C}}M^2_{v^*}\left[2\kappa_{\mathcal{F},w}\left(\frac{M_{\mathcal{C}}}{\mu_{\mathcal{F}}}\right)^2 + 3\right].\end{aligned} \tag{6.146}$$

*Assume further that* $\alpha, \beta$ *are set as*

$$\begin{aligned}\alpha &= 6\gamma\kappa_{\mathcal{F},w}\left(L_{\mathcal{U},1} + L^{w,\lambda}_{\mathcal{F},2}\left(4\kappa_{\mathcal{F},w}\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}} + \epsilon_{v,0}\right)\right)^2 + 1284\gamma\kappa^3_{\mathcal{F},w}M^2_{\mathcal{C}}M^2_{v^*} \\ \beta &= 3\gamma\kappa_{\mathcal{F},w}M^2_{\mathcal{C}}.\end{aligned} \tag{6.147}$$

*Then,* $\psi_{t+1} \leqslant \psi_t - \frac{\gamma}{8}\big\|\bar{D}_{\lambda,t}\big\|^2 - \frac{\alpha}{4\kappa_{\mathcal{F},w}}\|w_t - w^*_t\|^2$ *for all* $t \geqslant 0$.

*Proof.* By the implicit-descent and implicit-gradient-error bounds, we have

$$\begin{aligned}\phi(\lambda_{t+1}) &\leqslant \phi(\lambda_t) - \frac{\gamma}{8}\|\bar{D}_{\lambda,t}\|^2 - \frac{\gamma}{4}\|D_{\lambda,t}\|^2 + \frac{\gamma}{4}\|D_{\lambda,t} - \bar{D}_{\lambda,t}\|^2 + \frac{\gamma}{2}\big\|\nabla\phi(\lambda_t) - D^{\text{in}}_{\lambda,t}\big\|^2 + \frac{\gamma^2 L_\phi}{2}\|D_{\lambda,t}\|^2 \\ &\leqslant \phi(\lambda_t) - \frac{\gamma}{8}\|\bar{D}_{\lambda,t}\|^2 - \frac{\gamma}{4}(1 - 2\gamma L_\phi)\|D_{\lambda,t}\|^2 \\ &\quad + \frac{3\gamma}{2}\left(L_{\mathcal{U},1} + L^{w,\lambda}_{\mathcal{F},2}\left(\frac{4L_{\mathcal{U},0}}{\rho\mu^2_{\mathcal{F}}} + \epsilon_{v,0}\right)\right)^2\|w_t - w^*_t\|^2 + \frac{3\gamma}{2}M^2_{\mathcal{C}}\|v^*(\lambda_t) - v_t\|^2.\end{aligned} \tag{6.148}$$

Hence, using the lower-level drift and dual-error control bounds, we obtain,

$$
\begin{aligned}
\psi_{t+1} \leqslant \phi(\lambda_t) &- \frac{\gamma}{8}\|\bar{D}_{\lambda,t}\|^2 - \frac{\gamma}{4}(1-2\gamma L_\phi)\|D_{\lambda,t}\|^2 + \frac{3\gamma}{2}\left(L_{\mathcal{U},1} + L^{w,\lambda}_{\mathcal{F},2}\left(\frac{4L_{\mathcal{U},0}}{\rho\mu_{\mathcal{F}}^2} + \epsilon_{v,0}\right)\right)^2 \|w_t - w_t^*\|^2 \\
&+\frac{3\gamma}{2}M_{\mathcal{C}}^2\|v^*(\lambda_t) - v_t\|^2 + \alpha\left(\left(1 - \frac{\tau\mu_{\mathcal{F}}}{2}\right)\|w_t - w_t^*\|^2 + \frac{2}{\tau\mu_{\mathcal{F}}}\left(\frac{M_{\mathcal{C}}}{\mu_{\mathcal{F}}}\right)^2\gamma^2\|D_{\lambda,t}\|^2\right) \\
&+\beta\left(\left(1 - \frac{\rho\mu_{\mathcal{F}}}{2}\right)\|v_t - v^*(\lambda_t)\|^2 + \frac{107}{\rho\mu_{\mathcal{F}}}M_{v^*}^2\|w_t - w_t^*\|^2\right. \\
&\left.+\frac{25}{\rho\mu_{\mathcal{F}}}M_{v^*}^2\left[\frac{2}{\tau\mu_{\mathcal{F}}}\left(\frac{M_{\mathcal{C}}}{\mu_{\mathcal{F}}}\right)^2 + 3\right]\gamma^2\|D_{\lambda,t}\|^2\right) \\
= \psi_t &- \frac{\gamma}{8}\|\bar{D}_{\lambda,t}\|^2 - \alpha\frac{\tau\mu_{\mathcal{F}}}{4}\|w_t - w_t^*\|^2 - \frac{\gamma}{4}\left(1 - 2\gamma L_\phi - 8\gamma\frac{\alpha}{\tau\mu_{\mathcal{F}}}\left(\frac{M_{\mathcal{C}}}{\mu_{\mathcal{F}}}\right)^2\right. \\
&\left.+\gamma\beta\frac{100}{\rho\mu_{\mathcal{F}}}M_{v^*}^2\left[\frac{2}{\tau\mu_{\mathcal{F}}}\left(\frac{M_{\mathcal{C}}}{\mu_{\mathcal{F}}}\right)^2 + 3\right]\right)\|D_{\lambda,t}\|^2 \\
&-\left(\alpha\frac{\tau\mu_{\mathcal{F}}}{4} - \frac{3\gamma}{2}\left(L_{\mathcal{U},1} + L^{w,\lambda}_{\mathcal{F},2}\left(\frac{4L_{\mathcal{U},0}}{\rho\mu_{\mathcal{F}}^2} + \epsilon_{v,0}\right)\right)^2 - \beta\frac{107}{\rho\mu_{\mathcal{F}}}M_{v^*}^2\right)\|w_t - w_t^*\|^2 \\
&-\left(\beta\frac{\rho\mu_{\mathcal{F}}}{2} - \frac{3\gamma}{2}M_{\mathcal{C}}^2\right)\|v^*(\lambda_t) - v_t\|^2
\end{aligned}
\tag{6.149}
$$

Let us now set $\tau = \rho = \frac{1}{L^{w,w}_{\mathcal{F},1}}$, and

$$
\begin{aligned}
\alpha &= 6\gamma\kappa_{\mathcal{F},w}\left(L_{\mathcal{U},1} + L^{w,\lambda}_{\mathcal{F},2}\left(4\kappa_{\mathcal{F},w}\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}} + \epsilon_{v,0}\right)\right)^2 + 1284\gamma\kappa^3_{\mathcal{F},w}M_{\mathcal{C}}^2M_{v^*}^2 \\
\beta &= 3\gamma\kappa_{\mathcal{F},w}M_{\mathcal{C}}^2
\end{aligned}
\tag{6.150}
$$

where $\kappa_{\mathcal{F},w} = L^{w,w}_{\mathcal{F},1}/\mu_{\mathcal{F}}$ is the condition number of the lower-level problem with respect to $w$. This turns the previous inequality into

$$
\begin{aligned}
\psi_{t+1} \leqslant \psi_t &- \frac{\gamma}{8}\|\bar{D}_{\lambda,t}\|^2 - \frac{\alpha}{4\kappa_{\mathcal{F},w}}\|w_t - w_t^*\|^2 \\
&-\frac{\gamma}{4}\left(1 - 2\gamma L_\phi - 48\gamma^2\kappa^2_{\mathcal{F},w}\left(\frac{M_{\mathcal{C}}}{\mu_{\mathcal{F}}}\right)^2\left(L_{\mathcal{U},1} + L^{w,\lambda}_{\mathcal{F},2}\left(4\kappa_{\mathcal{F},w}\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}} + \epsilon_{v,0}\right)\right)^2\right. \\
&\quad -10272\gamma^2\kappa^4_{\mathcal{F},w}\frac{M_{\mathcal{C}}^4}{\mu_{\mathcal{F}}^2}M_{v^*}^2 \\
&\quad \left.+300\gamma^2\kappa^2_{\mathcal{F},w}M_{\mathcal{C}}^2M_{v^*}^2\left[2\kappa_{\mathcal{F},w}\left(\frac{M_{\mathcal{C}}}{\mu_{\mathcal{F}}}\right)^2 + 3\right]\right)\|D_{\lambda,t}\|^2
\end{aligned}
\tag{6.151}
$$

which simplifies to $\psi_{t+1} \leqslant \psi_t - \frac{\gamma}{8}\|\bar{D}_{\lambda,t}\|^2 - \frac{\alpha}{4\kappa_{\mathcal{F},w}}\|w_t - w_t^*\|^2$ as soon as $\gamma \leqslant \min\left(\frac{1}{\kappa_{\mathcal{F},w}}, \bar{\gamma}\right)$, where

$$\begin{aligned}\bar{\gamma}^{-1} = 2L_\phi + 48\kappa_{\mathcal{F},w}\left(\frac{M_\ell}{\mu_\mathcal{F}}\right)^2\left(L_{u,1} + L^{w,\lambda}_{\mathcal{F},2}\left(4\kappa_{\mathcal{F},w}\frac{L_{u,0}}{\mu_\mathcal{F}} + \epsilon_{v,0}\right)\right)^2 \\ +10272\kappa^3_{\mathcal{F},w}\frac{M^4_\ell}{\mu^2_\mathcal{F}}M^2_{v^*} \\ +300\kappa_{\mathcal{F},w}M^2_\ell M^2_{v^*}\left[2\kappa_{\mathcal{F},w}\left(\frac{M_\ell}{\mu_\mathcal{F}}\right)^2 + 3\right].\end{aligned} \tag{6.152}$$

□

Parameter policy for the stochastic setting

**Lemma 6.27.** *For any $t \geqslant 0$, it holds that*

$$\begin{aligned}\psi_{t+1} \leqslant \psi_t - \mathbb{E}\left[\frac{\gamma_t}{8}\|D_{\lambda,t}\|^2\right] - \mathbb{E}\left[\frac{\gamma_t}{8}\|\bar{D}_{\lambda,t}\|^2\right] - \frac{\alpha\tau\mu_\mathcal{F}}{4}\mathbb{E}\left[\|w_t - w^*_t\|^2\right] \\ -\mathbb{E}\left[\mathcal{C}_{D,t}\|D_{\lambda,t}\|^2\right] \\ -\mathbb{E}\left[\mathcal{C}_{w,t}\|w_t - w^*_t\|^2\right] \\ -\mathbb{E}\left[\mathcal{C}_{v,t}\|v_t - v^*(\lambda_t)\|^2\right] \\ -\mathbb{E}\left[\mathcal{C}_{w,v,t}\|w_t - w^*_t\|^2 \cdot \|v_t\|^2\right] \\ +\mathbb{E}\left[\mathcal{C}_{\psi,t}\right].\end{aligned} \tag{6.153}$$

*where the constants $\mathcal{C}_{D,t}, \mathcal{C}_{w,t}, \mathcal{C}_{v,t}, \mathcal{C}_{w,v,t}$ are defined as*

$$
\begin{aligned}
\mathcal{C}_{D,t} &\triangleq \frac{\gamma_t}{4}\Bigg(1 - 2\gamma_t L_\phi - 12\alpha\gamma_t\tau^{-1}\mu_{\mathcal{F}}^{-3}(M_e)^2 \\
&\quad - 108\beta\gamma_t\rho^{-1}\mu_{\mathcal{F}}^{-1}L_{v,*}\Big(1 + 3\tau^{-1}\mu_{\mathcal{F}}^{-3}M_{\bar{e}}^2\Big) \\
&\quad - 12\delta\gamma_t\tau^{-1}\mu_{\mathcal{F}}^{-3}M_{\bar{e}}^2\Bigg(\frac{15}{\rho\mu_{\mathcal{F}}}\Big(\frac{L_{u,0}}{\mu_{\mathcal{F}}}\Big)^2 + 2\rho^2\mathcal{V}_{u,1}^w\Bigg)\Bigg) \\
\mathcal{C}_{w,t} &\triangleq \alpha\frac{\tau\mu_{\mathcal{F}}}{4} - \frac{3\gamma}{2}\big(L_{u,1}\big)^2 - \beta\cdot 28\rho^{-1}\mu_{\mathcal{F}}^{-1}L_{v,*}\Big(2 + \tau^2\Big(L_{\mathcal{F},1}^{w,w}\Big)^2\Big) - \delta\Bigg(\frac{15}{\rho\mu_{\mathcal{F}}}\Big(\frac{L_{u,0}}{\mu_{\mathcal{F}}}\Big)^2 + 2\rho^2\mathcal{V}_{u,1}^w\Bigg) \\
\mathcal{C}_{v,t} &\triangleq \beta\frac{\rho\mu_{\mathcal{F}}}{2} - 3\gamma M_{\bar{e}}^2 \\
\mathcal{C}_{w,v,t} &\triangleq \delta\frac{\tau\mu_{\mathcal{F}}}{2} - 3\gamma\Big(L_{\mathcal{F},2}^{w,\lambda}\Big)^2 \\
\mathcal{C}_{\psi,t} &\triangleq 3\frac{\gamma}{4}\Big(\mathcal{V}_{u,1}^{\lambda} + \mathcal{V}_{\mathcal{F},2}^{w,\lambda}\mathcal{C}_{v,*}\Big) + \alpha\tau^2\frac{\kappa_{\mathcal{F},w}}{\kappa_{\mathcal{F},w}-1}\mathcal{V}_{\mathcal{F},1} \\
&\quad + \beta\Big(27\rho^{-1}\mu_{\mathcal{F}}^{-1}L_{v,*}\tau^2\frac{2\kappa_{\mathcal{F},w}}{\kappa_{\mathcal{F},w}-1}\mathcal{V}_{\mathcal{F},1} + 4\rho^2\Big(\mathcal{V}_{u,1}^w + \mathcal{V}_{\mathcal{F},2}^{w,w}\mathcal{C}_{v,*}\Big)\Big) \\
&\quad + \delta\Bigg(3\tau^{-1}\mu_{\mathcal{F}}^{-3}M_{\bar{e}}^2\mathbb{E}\Big[\mathbb{E}\Big[\big\|\gamma_t D_{\lambda,t}\big\|^2 \mid \mathfrak{S}_t\Big]\cdot\|v_t\|^2\Big] \\
&\qquad + \tau^2\frac{\kappa_{\mathcal{F},w}}{\kappa_{\mathcal{F},w}-1}\mathcal{V}_{\mathcal{F},1}\mathbb{E}\Big[\|v_t\|^2\Big] \\
&\qquad + \tau^2\Bigg(\frac{15}{\rho\mu_{\mathcal{F}}}\Big(\frac{L_{u,0}}{\mu_{\mathcal{F}}}\Big)^2 + 2\rho^2\mathcal{V}_{u,1}^w\Bigg)\frac{\kappa_{\mathcal{F},w}}{\kappa_{\mathcal{F},w}-1}\mathcal{V}_{\mathcal{F},1}\Bigg)
\end{aligned}
\tag{6.154}
$$

*Proof.* First observe that, following the exact same steps as in the proof of the implicit-descent bound, we have for any $t \geqslant 0$,

$$
\phi(\lambda_{t+1}) \leqslant \phi(\lambda_t) - \frac{\gamma_t}{8}\|\bar{D}_{\lambda,t}\|^2 - \frac{\gamma_t}{4}\|D_{\lambda,t}\|^2 + \frac{\gamma_t}{4}\|\bar{D}_{\lambda,t} - D_{\lambda,t}\|^2 + \frac{\gamma_t}{2}\Big\|\nabla\phi(\lambda_t) - D_{\lambda,t}^{\mathrm{in}}\Big\|^2 + \frac{\gamma_t^2 L_\phi}{2}\|D_{\lambda,t}\|^2. \tag{6.155}
$$

Thus, taking the expectation and using the stochastic implicit-gradient-error bound, we obtain

$$
\begin{aligned}
\mathbb{E}[\phi(\lambda_{t+1})] &\leqslant \mathbb{E}[\phi(\lambda_t)] - \mathbb{E}\Big[\frac{\gamma_t}{8}\|D_{\lambda,t}\|^2\Big] - \mathbb{E}\Big[\frac{\gamma_t}{8}\|\bar{D}_{\lambda,t}\|^2\Big] - \mathbb{E}\Big[\frac{\gamma_t}{8}(1 - 4\gamma_t L_\phi)\|D_{\lambda,t}\|^2\Big] \\
&\quad + \mathbb{E}\Big[\frac{3\gamma_t}{4}\Big\|\nabla\phi(\lambda_t) - D_{\lambda,t}^{\mathrm{in}}\Big\|^2\Big] \\
&\leqslant \mathbb{E}[\phi(\lambda_t)] - \mathbb{E}\Big[\frac{\gamma_t}{8}\|D_{\lambda,t}\|^2\Big] - \mathbb{E}\Big[\frac{\gamma_t}{8}\|\bar{D}_{\lambda,t}\|^2\Big] - \mathbb{E}\Big[\frac{\gamma_t}{8}(1 - 4\gamma_t L_\phi)\|D_{\lambda,t}\|^2\Big] \\
&\quad + \frac{3\gamma}{2}\big(L_{u,1}\big)^2\mathbb{E}\Big[\|w_t - w_t^*\|^2\Big] \\
&\quad + 3\gamma M_{\bar{e}}^2\mathbb{E}\Big[\|v_t - v^*(\lambda_t)\|^2\Big] \\
&\quad + 3\gamma\Big(L_{\mathcal{F},2}^{w,\lambda}\Big)^2\mathbb{E}\Big[\|w_t - w_t^*\|^2\cdot\|v_t\|^2\Big] \\
&\quad + \frac{3\gamma}{4}\Big(\mathcal{V}_{u,1}^{\lambda} + \mathcal{V}_{\mathcal{F},2}^{w,\lambda}\mathcal{C}_{v,*}\Big).
\end{aligned}
\tag{6.156}
$$

where we used that $\gamma_t \triangleq \gamma \min\big(1, \mathcal{C}_\gamma / \|D_{\lambda,t}\|\big) \leqslant \gamma$. Therefore, we may invoke Lemma 6.20, Lemma 6.23 and the expected-product bound to obtain

$$
\begin{aligned}
\psi_{t+1} \leqslant\ & \mathbb{E}[\phi(\lambda_t)] - \mathbb{E}\Big[\frac{\gamma_t}{8}\|D_{\lambda,t}\|^2\Big] - \mathbb{E}\Big[\frac{\gamma_t}{8}\|\bar{D}_{\lambda,t}\|^2\Big] - \mathbb{E}\Big[\frac{\gamma_t}{8}(1 - 4\gamma_t L_\phi)\|D_{\lambda,t}\|^2\Big] \\
& + \frac{3\gamma}{2}\big(L_{u,1}\big)^2 \mathbb{E}\Big[\|w_t - w_t^*\|^2\Big] \\
& + 3\gamma M_{\mathcal{C}}^2 \mathbb{E}\Big[\|\nu_t - \nu^*(\lambda_t)\|^2\Big] \\
& + 3\gamma\Big(L_{\mathcal{F},2}^{w,\lambda}\Big)^2 \mathbb{E}\Big[\|w_t - w_t^*\|^2 \cdot \|\nu_t\|^2\Big] \\
& + \frac{3\gamma}{4}\Big(\mathcal{V}_{u,1}^{\lambda} + \mathcal{V}_{\mathcal{F},2}^{w,\lambda}\mathcal{C}_{\nu,*}\Big) \\
& + \alpha\Bigg((1 - \tau\mu_{\mathcal{F}}/2)\mathbb{E}\Big[\|w_t - w_t^*\|^2\Big] + 3\tau^{-1}\mu_{\mathcal{F}}^{-3} M_{\mathcal{C}}^2 \mathbb{E}\Big[\big\|\gamma_t D_{\lambda,t}\big\|^2\Big] + \tau^2 \frac{\kappa_{\mathcal{F},w}}{\kappa_{\mathcal{F},w} - 1}\mathcal{V}_{\mathcal{F},1}\Bigg) \\
& + \beta\Bigg((1 - \rho\mu_{\mathcal{F}}/2)\mathbb{E}\Big[\|\nu_t - \nu^*(\lambda_t)\|^2\Big] + 28\rho^{-1}\mu_{\mathcal{F}}^{-1} L_{\nu,*}\Big(2 + \tau^2\Big(L_{\mathcal{F},1}^{w,w}\Big)^2\Big)\mathbb{E}\Big[\|w_t - w_t^*\|^2\Big] \\
& \quad + 27\rho^{-1}\mu_{\mathcal{F}}^{-1} L_{\nu,*}\Big(1 + 3\tau^{-1}\mu_{\mathcal{F}}^{-3} M_{\mathcal{C}}^2\Big)\mathbb{E}\Big[\big\|\gamma_t D_{\lambda,t}\big\|^2\Big] + 27\rho^{-1}\mu_{\mathcal{F}}^{-1} L_{\nu,*}\tau^2 \frac{2\kappa_{\mathcal{F},w}}{\kappa_{\mathcal{F},w} - 1}\mathcal{V}_{\mathcal{F},1} \\
& \quad + 4\rho^2\Big(\mathcal{V}_{u,1}^{w} + \mathcal{V}_{\mathcal{F},2}^{w,w}\mathcal{C}_{\nu,*}\Big)\Bigg) \\
& + \delta\Bigg(\Big(1 - \frac{\tau\mu_{\mathcal{F}}}{2}\Big)\mathbb{E}\Big[\|w_t - w_t^*\|^2 \cdot \|\nu_t\|^2\Big] + 3\tau^{-1}\mu_{\mathcal{F}}^{-3} M_{\mathcal{C}}^2 \mathbb{E}\Big[\mathbb{E}\Big[\big\|\gamma_t D_{\lambda,t}\big\|^2 \mid \mathfrak{S}_t\Big] \cdot \|\nu_t\|^2\Big] \\
& \quad + \tau^2 \frac{\kappa_{\mathcal{F},w}}{\kappa_{\mathcal{F},w} - 1}\mathcal{V}_{\mathcal{F},1}\mathbb{E}\Big[\|\nu_t\|^2\Big] + \Bigg(\frac{15}{\rho\mu_{\mathcal{F}}}\Big(\frac{L_{u,0}}{\mu_{\mathcal{F}}}\Big)^2 + 2\rho^2\mathcal{V}_{u,1}^{w}\Bigg)\mathbb{E}\Big[\|w_t - w_t^*\|^2\Big] \\
& \quad + 3\tau^{-1}\mu_{\mathcal{F}}^{-3} M_{\mathcal{C}}^2\Bigg(\frac{15}{\rho\mu_{\mathcal{F}}}\Big(\frac{L_{u,0}}{\mu_{\mathcal{F}}}\Big)^2 + 2\rho^2\mathcal{V}_{u,1}^{w}\Bigg)\mathbb{E}\Big[\mathbb{E}\Big[\big\|\gamma_t D_{\lambda,t}\big\|^2 \mid \mathfrak{S}_t\Big]\Big] \\
& \quad + \tau^2\Bigg(\frac{15}{\rho\mu_{\mathcal{F}}}\Big(\frac{L_{u,0}}{\mu_{\mathcal{F}}}\Big)^2 + 2\rho^2\mathcal{V}_{u,1}^{w}\Bigg)\frac{\kappa_{\mathcal{F},w}}{\kappa_{\mathcal{F},w} - 1}\mathcal{V}_{\mathcal{F},1}\Bigg).
\end{aligned}
\tag{6.157}
$$

where we used, in several places, $1 - \rho\mu_{\mathcal{F}}/8 \leqslant 1$ and $1 - \frac{\tau\mu_{\mathcal{F}}}{2} \leqslant 1$. After some rearrangement, we deduce that

$$
\begin{aligned}
\psi_{t+1} \leqslant\ & \psi_t - \mathbb{E}\Big[\frac{\gamma_t}{8}\big\|D_{\lambda,t}\big\|^2\Big] - \mathbb{E}\Big[\frac{\gamma_t}{8}\big\|\bar{D}_{\lambda,t}\big\|^2\Big] - \frac{\alpha\tau\mu_{\mathcal{F}}}{4}\mathbb{E}\Big[\|w_t - w_t^*\|^2\Big] \\
& - \mathbb{E}\Big[\mathcal{C}_{D,t}\big\|D_{\lambda,t}\big\|^2\Big] \\
& - \mathbb{E}\Big[\mathcal{C}_{w,t}\|w_t - w_t^*\|^2\Big] \\
& - \mathbb{E}\Big[\mathcal{C}_{\nu,t}\|\nu_t - \nu^*(\lambda_t)\|^2\Big] \\
& - \mathbb{E}\Big[\mathcal{C}_{w,\nu,t}\|w_t - w_t^*\|^2 \cdot \|\nu_t\|^2\Big] \\
& + \mathbb{E}\big[\mathcal{C}_{\psi,t}\big].
\end{aligned}
\tag{6.158}
$$

where the quantities $\mathcal{C}_{D,t}$, $\mathcal{C}_{w,t}$, $\mathcal{C}_{v,t}$, $\mathcal{C}_{w,v,t}$ and $\mathcal{C}_{\psi,t}$ are defined as

$$
\begin{aligned}
\mathcal{C}_{D,t} &\triangleq \frac{\gamma_t}{4}\Bigg(1 - 2\gamma_t L_\phi - 12\alpha\gamma_t\tau^{-1}\mu_{\mathcal{F}}^{-3}(M_{\mathcal{C}})^2 \\
&\quad -108\beta\gamma_t\rho^{-1}\mu_{\mathcal{F}}^{-1}L_{v,*}\Big(1 + 3\tau^{-1}\mu_{\mathcal{F}}^{-3}M_{\mathcal{C}}^2\Big) \\
&\quad -12\delta\gamma_t\tau^{-1}\mu_{\mathcal{F}}^{-3}M_{\mathcal{C}}^2\Bigg(\frac{15}{\rho\mu_{\mathcal{F}}}\Big(\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}}\Big)^2 + 2\rho^2\mathcal{V}_{\mathcal{U},1}^w\Bigg)\Bigg) \\
\mathcal{C}_{w,t} &\triangleq \alpha\frac{\tau\mu_{\mathcal{F}}}{4} - \frac{3\gamma}{2}(L_{\mathcal{U},1})^2 - \beta\cdot 28\rho^{-1}\mu_{\mathcal{F}}^{-1}L_{v,*}\Big(2 + \tau^2\Big(L_{\mathcal{F},1}^{w,w}\Big)^2\Big) - \delta\Bigg(\frac{15}{\rho\mu_{\mathcal{F}}}\Big(\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}}\Big)^2 + 2\rho^2\mathcal{V}_{\mathcal{U},1}^w\Bigg) \\
\mathcal{C}_{v,t} &\triangleq \beta\frac{\rho\mu_{\mathcal{F}}}{2} - 3\gamma M_{\mathcal{C}}^2 \\
\mathcal{C}_{w,v,t} &\triangleq \delta\frac{\tau\mu_{\mathcal{F}}}{2} - 3\gamma\Big(L_{\mathcal{F},2}^{w,\lambda}\Big)^2 \\
\mathcal{C}_{\psi,t} &\triangleq 3\frac{\gamma}{4}\Big(\mathcal{V}_{\mathcal{U},1}^{\lambda} + \mathcal{V}_{\mathcal{F},2}^{w,\lambda}\mathcal{C}_{v,*}\Big) + \alpha\tau^2\frac{\kappa_{\mathcal{F},w}}{\kappa_{\mathcal{F},w}-1}\mathcal{V}_{\mathcal{F},1} \\
&\quad +\beta\Big(27\rho^{-1}\mu_{\mathcal{F}}^{-1}L_{v,*}\tau^2\frac{2\kappa_{\mathcal{F},w}}{\kappa_{\mathcal{F},w}-1}\mathcal{V}_{\mathcal{F},1} + 4\rho^2\Big(\mathcal{V}_{\mathcal{U},1}^w + \mathcal{V}_{\mathcal{F},2}^{w,w}\mathcal{C}_{v,*}\Big)\Big) \\
&\quad +\delta\Bigg(3\tau^{-1}\mu_{\mathcal{F}}^{-3}M_{\mathcal{C}}^2\mathbb{E}\Big[\mathbb{E}\Big[\big\|\gamma_t D_{\lambda,t}\big\|^2 \mid \mathfrak{S}_t\Big]\cdot\|v_t\|^2\Big] \\
&\quad\ +\tau^2\frac{\kappa_{\mathcal{F},w}}{\kappa_{\mathcal{F},w}-1}\mathcal{V}_{\mathcal{F},1}\mathbb{E}\Big[\|v_t\|^2\Big] \\
&\quad\ +\tau^2\Bigg(\frac{15}{\rho\mu_{\mathcal{F}}}\Big(\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}}\Big)^2 + 2\rho^2\mathcal{V}_{\mathcal{U},1}^w\Bigg)\frac{\kappa_{\mathcal{F},w}}{\kappa_{\mathcal{F},w}-1}\mathcal{V}_{\mathcal{F},1}\Bigg)
\end{aligned}
\tag{6.159}
$$

□

**Lemma 6.28.** *Assume that the step sizes $\tau$ and $\rho$ are set as*

$$
\tau = 1/L_{\mathcal{F},1}^{w,w},\ \rho = \min\Big(1/L_{\mathcal{F},1}^{w,w},\ \mu_{\mathcal{F}}/\Big(4\mathcal{V}_{\mathcal{F},2}^{w,w}\Big)\Big) \tag{6.160}
$$

*and $\gamma$ is set as $\gamma = \min\Big(\bar{\gamma}, L_\phi^{-1}\Big)$, where*

$$
\begin{aligned}
\bar{\gamma}^{-1} &\triangleq 2L_\phi + 72M_{\mathcal{C}}^2\kappa_{\mathcal{F},w}^2(L_{\mathcal{U},1})^2L_\phi^{-1}\mu_{\mathcal{F}}^{-2} \\
&\quad +32256L_{v,*}M_{\mathcal{C}}^4\kappa_{\mathcal{F},w}^2L_\phi^{-1}\mu_{\mathcal{F}}^{-2}\Big(\kappa_{\mathcal{F},w} + 4\mu_{\mathcal{F}}^{-2}\mathcal{V}_{\mathcal{U},1}^w\Big)^2 \\
&\quad +288\kappa_{\mathcal{F},w}^2M_{\mathcal{C}}^2\Big(L_{\mathcal{F},2}^{w,w}\Big)^2L_\phi^{-1}\mu_{\mathcal{F}}^{-2}\Bigg(15\Big(\kappa_{\mathcal{F},w} + 4\mu_{\mathcal{F}}^{-2}\mathcal{V}_{\mathcal{U},1}^w\Big)\Big(\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}}\Big)^2 + 2\frac{\mathcal{V}_{\mathcal{U},1}^w}{\Big(L_{\mathcal{F},1}^{w,w}\Big)^2}\Bigg) \\
&\quad +648L_{v,*}M_{\mathcal{C}}^2L_\phi^{-1}\Big(1 + 3\kappa_{\mathcal{F},w}\mu_{\mathcal{F}}^{-2}M_{\mathcal{C}}^2\Big)\Big(\kappa_{\mathcal{F},w} + 4\mu_{\mathcal{F}}^{-2}\mathcal{V}_{\mathcal{U},1}^w\Big)^2 \\
&\quad +72\kappa_{\mathcal{F},w}^2\Big(L_{\mathcal{F},2}^{w,w}\Big)^2M_{\mathcal{C}}^2L_\phi^{-1}\mu_{\mathcal{F}}^{-2}\Bigg(15\Big(\kappa_{\mathcal{F},w} + 4\mu_{\mathcal{F}}^{-2}\mathcal{V}_{\mathcal{U},1}^w\Big)\Big(\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}}\Big)^2 + 2\frac{\mathcal{V}_{\mathcal{U},1}^w}{\Big(L_{\mathcal{F},1}^{w,w}\Big)^2}\Bigg)
\end{aligned}
\tag{6.161}
$$

*Furthermore, assume that the Lyapunov parameters $\alpha$, $\beta$, and $\delta$ are set as:*

$$\alpha = \gamma\Bigg( 6\kappa_{\mathcal{F},w}\big(L_{\mathcal{U},1}\big)^2 + 2688\kappa_{\mathcal{F},w}\Big(\kappa_{\mathcal{F},w} + 4\mu_{\mathcal{F}}^{-2}\nu_{\mathcal{U},1}^{w}\Big)^2 L_{v,*} M_{\mathcal{C}}^2 \\ +24\kappa_{\mathcal{F},w}\Big(L_{\mathcal{F},2}^{w,w}\Big)^2\Bigg( 15\Big(\kappa_{\mathcal{F},w} + 4\mu_{\mathcal{F}}^{-2}\nu_{\mathcal{U},1}^{w}\Big)\Big(\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}}\Big)^2 + 2\frac{\nu_{\mathcal{U},1}^{w}}{\Big(L_{\mathcal{F},1}^{w,w}\Big)^2}\Bigg)\Bigg) \tag{6.162}$$

$$\beta = 6\gamma\Big(\kappa_{\mathcal{F},w} + 4\mu_{\mathcal{F}}^{-2}\nu_{\mathcal{U},1}^{w}\Big)M_{\mathcal{C}}^2, \quad \delta = 6\gamma\kappa_{\mathcal{F},w}\Big(L_{\mathcal{F},2}^{w,w}\Big)^2.$$

*Then, the constants $C_{D,t}, C_{w,t}, C_{v,t}, C_{w,v,t}$ defined in Lemma 6.27 are all nonnegative.*

*Proof.* It suffices to plug the expressions given for $\tau$, $\rho$, $\gamma$, $\alpha$, $\beta$, and $\delta$ into the definitions of the constants $C_{D,t}, C_{w,t}, C_{v,t}, C_{w,v,t}$ given in Lemma 6.27. One should begin with $C_{v,t}$ and $C_{w,v,t}$, then $C_{w,t}$, and finally $C_{D,t}$. □

**Lemma 6.29.** *Under this parameterization given in Lemma 6.28, the remaining term $\mathbb{E}\big[C_{\psi,t}\big]$ may be upper bounded as*

$$\begin{aligned}\mathbb{E}\big[C_{\psi,t}\big] \leqslant C_{\psi} \triangleq \gamma\Bigg[ & \frac{3}{4}\Big(\nu_{\mathcal{U},1}^{\lambda} + \nu_{\mathcal{F},2}^{w,\lambda} C_{v,*}\Big) \\ & +6\big(L_{\mathcal{U},1}\big)^2\big(\kappa_{\mathcal{F},w} - 1\big)^{-1}\mu_{\mathcal{F}}^{-2}\nu_{\mathcal{F},1} \\ & +2688 L_{v,*}\Big(\kappa_{\mathcal{F},w} + 4\mu_{\mathcal{F}}^{-2}\nu_{\mathcal{U},1}^{w}\Big)^2 M_{\mathcal{C}}^2\big(\kappa_{\mathcal{F},w} - 1\big)^{-1}\mu_{\mathcal{F}}^{-2}\nu_{\mathcal{F},1} \\ & +24\frac{\kappa_{\mathcal{F},w}^2}{\kappa_{\mathcal{F},w} - 1}\Bigg( 15\Big(\kappa_{\mathcal{F},w} + 4\mu_{\mathcal{F}}^{-2}\nu_{\mathcal{U},1}^{w}\Big)\Big(\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}}\Big)^2 + 2\frac{\nu_{\mathcal{U},1}^{w}}{\Big(L_{\mathcal{F},1}^{w,w}\Big)^2}\Bigg)\nu_{\mathcal{F},1} \\ & +324 L_{v,*} M_{\mathcal{C}}^2\Big(\kappa_{\mathcal{F},w} + 4\mu_{\mathcal{F}}^{-2}\nu_{\mathcal{U},1}^{w}\Big)^2\big(\kappa_{\mathcal{F},w} - 1\big)^{-1}\mu_{\mathcal{F}}^{-1}\Big(L_{\mathcal{F},1}^{w,w}\Big)^{-1}\nu_{\mathcal{F},1} \\ & +24\Big(L_{\mathcal{F},1}^{w,w}\Big)^{-1}\mu_{\mathcal{F}}^{-1} M_{\mathcal{C}}^2\Big(\nu_{\mathcal{U},1}^{w} + \nu_{\mathcal{F},2}^{w,w} C_{v,*}\Big) \\ & +6 C_{v,*}\Big(L_{\mathcal{F},2}^{w,w}\Big)^2\big(\kappa_{\mathcal{F},w} - 1\big)^{-1}\mu_{\mathcal{F}}^{-2}\nu_{\mathcal{F},1} \\ & +6\Big(L_{\mathcal{F},2}^{w,w}\Big)^2\mu_{\mathcal{F}}^{-2}\Bigg( 15\Big(\kappa_{\mathcal{F},w} + 4\mu_{\mathcal{F}}^{-2}\nu_{\mathcal{U},1}^{w}\Big)\Big(\frac{L_{\mathcal{U},0}}{\mu_{\mathcal{F}}}\Big)^2 + 2\frac{\nu_{\mathcal{U},1}^{w}}{\Big(L_{\mathcal{F},1}^{w,w}\Big)^2}\Bigg)\big(\kappa_{\mathcal{F},w} - 1\big)^{-1}\nu_{\mathcal{F},1}\Bigg] \\ & +18\gamma^3\Big(L_{\mathcal{F},2}^{w,w}\Big)^2\mu_{\mathcal{F}}^{-4} M_{\mathcal{C}}^2 C_{v,*} C_{\Upsilon}^2\end{aligned} \tag{6.163}$$

*Proof.* This again follows directly by substituting the parameterization given in Lemma 6.28 into the definition of $C_{\psi,t}$ given in Lemma 6.27. □

# 7 CONCLUSIONS

In this section, we discuss the conclusions of the different projects presented in this thesis, and we identify open questions and perspectives for future work.

In Chapter 3, we addressed the cost of training kernel support vector machines on large datasets. We propose the `snacks` solver, which combines two known techniques, a Nyström approximation and a restarted accelerated stochastic subgradient method, applied to the nonsmooth primal formulation. We analyzed the computational cost of the solver and discussed its extension to other losses and regularizers. We openly released the `snacks` toolbox, which uses a simple `scikit-learn` interface for users to train kernel SVMs on large datasets. We provided numerical experiments showing competitive predictive accuracy and shorter training times, with an advantage that grew with dataset size.

In Chapter 4, we identified dual methods that compute convex hull prices efficiently on large electricity market instances. We reviewed the pricing problem and its standard formulations, and provided the `ConvexHullPricing.jl` toolbox to implement and benchmark methods for computing these prices. Our experiments on Belgian and Californian instances showed that carefully initialized subgradient methods were competitive, while variable-metric multi-cut bundle methods gave the best overall performance. These comparisons explain the benefit of retaining the separable structure of the Lagrangian and provide practical guidance on method selection.

In Chapter 5, we proposed a modeling language to help select first-order methods for a given optimization problem. Our `Argo.jl` package makes it possible to compare methods across supported reformulations using their worst-case convergence guarantees. Each recommendation comes with a certificate explaining why the method applies and how its guarantee transfers to the original problem. We also used these guarantees to choose when to switch between methods and to certify methods for solving proximal subproblems approximately. Our numerical examples illustrate both the usefulness of these guarantees and their limits as predictors of practical performance.

Finally, in Chapter 6, we studied how to improve a chosen fairness metric while preserving Pareto efficiency across groups. The underlying BADR formulation learns

the group weights through a bilevel problem, for which we developed deterministic and stochastic single-loop algorithms. Our convergence analysis accounts for the lack of global smoothness caused by the coupling between the weights and group losses. We provided numerical experiments showing improvements over existing Pareto-efficient approaches and examined how these gains affect predictive performance on test data. We also released the `badr` toolbox to make our approach easy to apply on several learning tasks and for various fairness criteria.

## 7.1 Open questions

We conclude by identifying a handful of open questions, some fundamental and other incremental, raised by preceding chapters and that we believe are worth pursuing in future work.

**Representing and searching reformulations.** We first ask how useful reformulations can be generated automatically. `Argo.jl` provides a first representation through structured objectives and registered reformulation rules whose preservation properties are recorded. Its present library contains only simple operations, such as regrouping terms and transferring curvature. Extending it to more involved transformations such as dualization would allow for a broader search of the space of reformulations. Such transformations are often regarded as too problem-specific to be automated. We argue that almost all reformulation techniques can be formalized, even involved ones, while searching for the subexpression on which to apply them, and with which parameters, remains a matter of expertise. This search can nevertheless be automated in some cases, as in the Risch algorithm for the choice of substitution in symbolic integration (Risch, 1969) and, to a lesser extent, in `Argo.jl` for the curvature transfer parameter $\rho$. Identifying the classes of problems on which this search can be automated remains open.

**Generation of optimization methods.** Currently, `Argo.jl` can generate new methods only by combining known methods with known reformulations, but it is conceivable that the modeling language and calculus could be extended to generate new methods automatically. Most first-order optimization methods fall into already known algorithmic frameworks: successive outer approximation of the solution set, successive minimization of a locally tight approximation, etc. It would be interesting to explore whether the modeling language and calculus of `Argo.jl` could be extended to generate new methods automatically, for instance by automatically deriving a Taylor polynomial approximation of the objective and minimizing it at each iteration, such as in Streeter (2023).

**Ranking formulation–method pairs.** We then ask how the generated candidates should be compared. `Argo.jl` ranks methods by worst-case convergence guarantees established for each formulation, but these guarantees provide only a coarse estimate of running time. In particular, it should be possible to apply techniques from the *per-instance algorithm selection* literature (Kerschke et al., 2019) to build genuine recommender systems for optimization methods. This requires a large enough database of problem instances to train the model, and to identify features of the problem that are predictive of method performance. The latter can be challenging, as the features must be computable without solving the problem itself while the former has yet to be built for the problem classes considered in this thesis[18].

**Improving bilevel optimization methods.** While working on `badr`, we were struck by the lack of convergent algorithms for our bilevel framework and therefore took significant care in developing efficient single-loop methods with both theoretical and empirical guarantees. Still, the current algorithm only converges to a noise-dominated neighbourhood of a stationary point and does not exploit finite-sum structure. A first step to improve the algorithm would be a variance-reduced alternative to the current BADR-SGD, for example based on the approach of Dagréou et al. (2022).

**Extending adaptive rescalarization beyond fairness.** Although BADR was primarily motivated by fairness in machine learning, it is in fact a general approach, applicable to any multi-objective optimization problem for which a preference metric can be defined on the Pareto frontier. A first step would be to extend the framework to settings beyond fairness. For that, there are plenty of problems in machine learning and operations research that can be formulated as multi-objective optimization problems, for which I believe the bilevel adaptive rescalarization approach could be useful. Such extensions would naturally raise algorithmic questions, since the lower-level problems are usually not strongly convex, making the convergence analysis more challenging.

[18] This is in contrast with linear programs and mixed-integer programs which have more established and diverse benchmarking instances.

# BIBLIOGRAPHY


Abernethy, J., Awasthi, P., Kleindessner, M., Morgenstern, J., Russell, C., & Zhang, J. (2020). Active sampling for min-max fairness. *Arxiv Preprint Arxiv:2006.06879*.

Agrawal, A., Verschueren, R., Diamond, S., & Boyd, S. (2018). A rewriting system for convex optimization problems. *Journal of Control and Decision*, *5*(1), 42–60.

Andrianesis, P., Bertsimas, D., Caramanis, M. C., & Hogan, W. W. (2021). Computation of convex hull prices in electricity markets with non-convexities using Dantzig-Wolfe decomposition. *IEEE Transactions on Power Systems*, *37*(4), 2578–2589.

Arbel, M., & Mairal, J. (2021). Amortized implicit differentiation for stochastic bilevel optimization. *Arxiv Preprint Arxiv:2111.14580*.

Aronszajn, N. (1950). Theory of reproducing kernels. *Transactions of the American Mathematical Society*, *68*(3), 337–404.

Ayan, B., Abacıoğlu, S., & Basilio, M. P. (2023). A comprehensive review of the novel weighting methods for multi-criteria decision-making. *Information*, *14*(5), 285.

Bartz-Beielstein, T., Doerr, C., Berg, D. van den, Bossek, J., Chandrasekaran, S., Eftimov, T., Fischbach, A., Kerschke, P., La Cava, W., Lopez-Ibanez, M., Malan, K. M., Moore, J. H., Naujoks, B., Orzechowski, P., Volz, V., Wagner, M., & Weise, T. (2020). Benchmarking in optimization: Best practice and open issues. *Arxiv Preprint Arxiv:2007.03488*.

Beck, A. (2017). *First-order methods in optimization*. SIAM.

Beck, A., & Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, *2*(1), 183–202.

Becker, S. R., Candès, E. J., & Grant, M. C. (2011). Templates for convex cone problems with applications to sparse signal recovery. *Mathematical Programming Computation*, *3*, 165–218.

Berk, R., Heidari, H., Jabbari, S., Joseph, M., Kearns, M., Morgenstern, J., Neel, S., & Roth, A. (2017). A convex framework for fair regression. *Arxiv Preprint Arxiv:1706.02409*.

Bertrand, Q., Klopfenstein, Q., Massias, M., Blondel, M., Vaiter, S., Gramfort, A., & Salmon, J. (2022). Implicit differentiation for fast hyperparameter selection in non-smooth convex learning. *Journal of Machine Learning Research*, *23*(149), 1–43.

Beznosikov, A., Polyak, B., Gorbunov, E., Kovalev, D., & Gasnikov, A. (2023). Smooth monotone stochastic variational inequalities and saddle point problems: A survey. *European Mathematical Society Magazine*, 15–28.

Blodgett, S. L., Green, L., & O'Connor, B. (2016). Demographic dialectal variation in social media: A case study of African-American English. *Arxiv Preprint Arxiv:1608.08868*.

Boyd, S. P., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Boyd, S., Xiao, L., & Mutapcic, A. (2003). Subgradient methods. *Lecture Notes of Ee392o, Stanford University, Autumn Quarter*, *2004*, 2004–2005.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., & Zhang, Q. (2018,). *JAX: composable transformations of Python+NumPy programs*. http://github.com/jax-ml/jax

Branke, J. (2008). *Multiobjective optimization: Interactive and evolutionary approaches* (Vol. 5252). Springer Science & Business Media.

Braun, G., Carderera, A., Combettes, C. W., Hassani, H., Karbasi, A., Mokhtari, A., & Pokutta, S. (2022). Conditional gradient methods. *Arxiv Preprint Arxiv:2211.14103*.

Briant, O., Lemaréchal, C., Meurdesoif, P., Michel, S., Perrot, N., & Vanderbeck, F. (2008). Comparison of bundle and classical column generation. *Mathematical Programming*, *113*, 299–344.

Bussieck, M. R., & Meeraus, A. (2004). General algebraic modeling system (GAMS). In *Modeling languages in mathematical optimization: Modeling languages in mathematical optimization* (pp. 137–157). Springer.

Byers, C., & Hug, G. (2023). Long-run optimal pricing in electricity markets with non-convex costs. *European Journal of Operational Research*, *307*(1), 351–363.

Calders, T., Kamiran, F., & Pechenizkiy, M. (2009). Building classifiers with independency constraints. *2009 IEEE International Conference on Data Mining Workshops*, 13–18.

Calmon, F., Wei, D., Vinzamuri, B., Natesan Ramamurthy, K., & Varshney, K. R. (2017). Optimized pre-processing for discrimination prevention. *Advances in Neural Information Processing Systems*, *30*.

Caton, S., & Haas, C. (2024). Fairness in Machine Learning: A Survey. *ACM Computing Surveys*, *56*(7), 1–38. https://doi.org/10.1145/3616865

Cederberg, D., Zhang, W., Nobel, P., & Boyd, S. (2026,). *Disciplined Nonlinear Programming*. https://arxiv.org/abs/2606.02896

Celis, L. E., Huang, L., Keswani, V., & Vishnoi, N. K. (2019). Classification with fairness constraints: A meta-algorithm with provable guarantees. *Proceedings of the Conference on Fairness, Accountability, And Transparency*, 319–328.

Chang, C.-C., & Lin, C.-J. (2011). LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *2*(3), 1–27.

Chayti, E. M., & Jaggi, M. (2024). A new first-order meta-learning algorithm with convergence guarantees. *Arxiv Preprint Arxiv:2409.03682*.

Chen, L., & Zhang, J. (2025). On the Condition Number Dependency in Bilevel Optimization. *Arxiv Preprint Arxiv:2511.22331*.

Chen, T., Sun, Y., Xiao, Q., & Yin, W. (2022). A single-timescale method for stochastic bilevel optimization. *International Conference on Artificial Intelligence and Statistics*, 2466–2488.

Chen, Y., Epperly, E. N., Tropp, J. A., & Webber, R. J. (2025). Randomly pivoted Cholesky: Practical approximation of a kernel matrix with few entry evaluations. *Communications on Pure and Applied Mathematics*, *78*(5), 995–1041.

Citron, D. K., & Pasquale, F. (2014). The scored society: Due process for automated predictions. *Wash. L. Rev.*, *89*, 1.

Coello, C. A. C., Pulido, G. T., & Lechuga, M. S. (2004). Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, *8*(3), 256–279.

Conforti, M., Cornuéjols, G., Zambelli, G., Conforti, M., Cornuéjols, G., & Zambelli, G. (2014). *Integer programming models*. Springer.

Conn, A. R., Gould, N. I., & Toint, P. L. (2000). *Trust region methods*. SIAM.

Cotter, A., Jiang, H., Gupta, M., Wang, S., Narayan, T., You, S., & Sridharan, K. (2019). Optimization with non-differentiable constraints with applications to fairness, recall, churn, and other goals. *Journal of Machine Learning Research*, *20*(172), 1–59.

Cross, J. L., Choma, M. A., & Onofrey, J. A. (2024). Bias in medical AI: Implications for clinical decision-making. *PLOS Digital Health*, *3*(11), e0000651.

Cruz, A. F., Belém, C., Jesus, S., Bravo, J., Saleiro, P., & Bizarro, P. (2022). FairGBM: Gradient boosting with fairness constraints. *Arxiv Preprint Arxiv:2209.07850*.

Cutkosky, A., & Boahen, K. (2017). Online learning without prior information. *Conference on Learning Theory*, 643–677.

Dagréou, M., Ablin, P., Vaiter, S., & Moreau, T. (2022). A framework for bilevel optimization that enables stochastic and global variance reduction algorithms. *Advances in Neural Information Processing Systems*, *35*, 26698–26710.

Das, I., & Dennis, J. E. (1997). A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems. *Structural Optimization*, *14*(1), 63–69.

Davis, D., & Yin, W. (2017). A three-operator splitting scheme and its optimization applications. *Set-Valued and Variational Analysis*, *25*(4), 829–858. https://doi.org/10.1007/s11228-017-0421-z

Defazio, A., & Mishchenko, K. (2023). Learning-rate-free learning by D-Adaptation. *International Conference on Machine Learning*, 7449–7479.

Della Vecchia, A., De Vito, E., Mourtada, J., & Rosasco, L. (2024). The Nyström method for convex loss functions. *Journal of Machine Learning Research*, *25*(360), 1–60.

Delogne, R., & Jacques, L. (2025). Random Features for Grassmannian Kernels. *2025 IEEE Statistical Signal Processing Workshop (SSP)*, 1–4. https://doi.org/10.1109/SSP64130.2025.11073301

Dempe, S. (2002). *Foundations of bilevel programming*. Springer.

Dempe, S., & Zemkoho, A. (2020). *Bilevel Optimization: Advances and Next Challenges* (Vol. 161). Springer Nature.

Devolder, O., Glineur, F., & Nesterov, Y. (2014). First-order methods of smooth convex optimization with inexact oracle. *Mathematical Programming*, *146*, 37–75.

Diamond, S., & Boyd, S. (2016). CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, *17*(83), 1–5.

Diana, E., Gill, W., Kearns, M., Kenthapadi, K., & Roth, A. (2021). Minimax group fairness: Algorithms and experiments. *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, And Society*, 66–76.

Ding, F., Hardt, M., Miller, J., & Schmidt, L. (2021). Retiring Adult: New Datasets for Fair Machine Learning. *Advances in Neural Information Processing Systems*, *34*.

Do, H., Putzel, P., Martin, A. S., Smyth, P., & Zhong, J. (2022). Fair generalized linear models with a convex penalty. *International Conference on Machine Learning*, 5286–5308.

Domke, J. (2012). Generic methods for optimization-based modeling. *Artificial Intelligence and Statistics*, 318–326.

Donini, M., Oneto, L., Ben-David, S., Shawe-Taylor, J. S., & Pontil, M. (2018). Empirical risk minimization under fairness constraints. *Advances in Neural Information Processing Systems*, *31*.

Dontchev, A. L., & Rockafellar, R. T. (2014). *Implicit Functions and Solution Mappings* (2nd ed., p. xxviii+466). Springer. https://doi.org/10.1007/978-1-4939-1037-3

Drori, Y., & Teboulle, M. (2014). Performance of first-order methods for smooth convex minimization: a novel approach. *Mathematical Programming*, *145*(1–2), 451–482. https://doi.org/10.1007/s10107-013-0653-0

Dunning, I., Huchette, J., & Lubin, M. (2017). JuMP: A Modeling Language for Mathematical Optimization. *SIAM Review*, *59*(2), 295–320. https://doi.org/10.1137/15M1020575

Dwork, C., Hardt, M., Pitassi, T., Reingold, O., & Zemel, R. (2012). Fairness through awareness. *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, 214–226.

Ehrgott, M. (2005). *Multicriteria optimization*. Springer.

Eichfelder, G. (2008). *Adaptive scalarization methods in multiobjective optimization*. Springer.

Epperly, E. N., Tropp, J. A., & Webber, R. J. (2024). Xtrace: Making the most of every sample in stochastic trace estimation. *SIAM Journal on Matrix Analysis and Applications*, *45*(1), 1–23.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., & Lin, C.-J. (2008). LIBLINEAR: A library for large linear classification. *The Journal of Machine Learning Research*, *9*, 1871–1874.

Feldman, M., Friedler, S. A., Moeller, J., Scheidegger, C., & Venkatasubramanian, S. (2015). Certifying and removing disparate impact. *Proceedings of the 21th ACM*

*SIGKDD International Conference on Knowledge Discovery and Data Mining*, 259–268.

Fenchel, W., & Blackett, D. W. (1953). *Convex cones, sets, and functions*. Princeton University, Department of Mathematics, Logistics Research Project.

Fernández-Blanco, R., Arroyo, J. M., & Alguacil, N. (2015). Bilevel programming for price-based electricity auctions: A revenue-constrained case. *EURO Journal on Computational Optimization*, *3*(3), 163–195.

Fine, S., & Scheinberg, K. (2001). Efficient SVM training using low-rank kernel representations. *Journal of Machine Learning Research*, *2*, 243–264.

Fourer, R., Gay, D. M., & Kernighan, B. W. (1990). AMPL: A mathematical programming language. *Management Science*, *36*(5), 519–554.

Frangioni, A. (2020). Standard bundle methods: untrusted models and duality. *Numerical Nonsmooth Optimization: State of the Art Algorithms*, 61–116.

Gass, S., & Saaty, T. (1955). The computational algorithm for the parametric objective function. *Naval Research Logistics Quarterly*, *2*(1–2), 39–45.

Geoffrion, A. M. (1968). Proper efficiency and the theory of vector maximization. *Journal of Mathematical Analysis and Applications*, *22*(3), 618–630.

Geoffrion, A. M. (1974). Lagrangian relaxation for integer programming. *Mathematical Programming Study*, *2*, 82–114. https://doi.org/10.1007/BFb0120690

Ghadimi, S., & Wang, M. (2018). Approximation methods for bilevel programming. *Arxiv Preprint Arxiv:1802.02246*.

Ghadimi, S., Lan, G., & Zhang, H. (2016). Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization. *Mathematical Programming*, *155*(1), 267–305.

GN Hearing AS. (2017). *Performance based in situ optimization of hearing aids* (Patent No. 9838805B2). https://patents.google.com/patent/US10154357B2/

Gong, X., Hao, J., & Liu, M. (2024). An accelerated algorithm for stochastic bilevel optimization under unbounded smoothness. *Advances in Neural Information Processing Systems*, *37*, 78201–78243.

Goujaud, B., Moucer, C., Glineur, F., Hendrickx, J., Taylor, A. B., & Dieuleveut, A. (2024). PEPit: computer-assisted worst-case analyses of first-order optimization methods in Python. *Mathematical Programming Computation*, *16*(3), 337–367. https://doi.org/10.1007/s12532-024-00259-7

Goulart, P. J., & Chen, Y. (2024,). *Clarabel: An interior-point solver for conic programs with quadratic objectives*.

Grant, M., & Boyd, S. (2014,). *CVX: Matlab software for disciplined convex programming, version 2.1*.

Gretton, A., Bousquet, O., Smola, A., & Schölkopf, B. (2005). Measuring statistical dependence with Hilbert-Schmidt norms. *International Conference on Algorithmic Learning Theory*, 63–77.

Gribik, P. R., Hogan, W. W., & Pope, S. L. (2007). Market-clearing electricity prices and energy uplift. *Cambridge, MA*, 1–46.

Grimmer, B. (2019). Convergence rates for deterministic and stochastic subgradient methods without Lipschitz continuity. *SIAM Journal on Optimization*, *29*(2), 1350–1365. https://doi.org/10.1137/18M117306X

Güler, O. (1992). New Proximal Point Algorithms for Convex Minimization. *SIAM Journal on Optimization*, *2*(4), 649–664. https://doi.org/10.1137/0802032

Hardt, M., Price, E., & Srebro, N. (2016). Equality of opportunity in supervised learning. *Advances in Neural Information Processing Systems*, *29*.

Hashimoto, T., Srivastava, M., Namkoong, H., & Liang, P. (2018). Fairness without demographics in repeated loss minimization. *International Conference on Machine Learning*, 1929–1938.

Hogan, W. W., & Ring, B. J. (2003). On minimum-uplift pricing for electricity markets. *Electricity Policy Group*, 1–30.

Hsieh, C.-J., Chang, K.-W., Lin, C.-J., Keerthi, S. S., & Sundararajan, S. (2008). A dual coordinate descent method for large-scale linear SVM. *Proceedings of the 25th International Conference on Machine Learning*, 408–415.

Hua, B., & Baldick, R. (2016). A convex primal formulation for convex hull pricing. *IEEE Transactions on Power Systems*, *32*(5), 3814–3823.

Huawei Technologies Co Ltd. (2022). *Image Processing Method and Electronic Device* (Patent No. US20220180485A1). https://patents.google.com/patent/US20220180485A1

Ito, N., Takeda, A., & Namerikawa, T. (2013). Convex hull pricing for demand response in electricity markets. *2013 IEEE International Conference on Smart Grid Communications (Smartgridcomm)*, 151–156.

Iutzeler, F., Malick, J., & Oliveira, W. de. (2020). Asynchronous level bundle methods. *Mathematical Programming*, *184*(1), 319–348. https://doi.org/10.1007/s10107-019-01414-y

Jalving, J., Shin, S., & Zavala, V. M. (2022). A graph-based modeling abstraction for optimization: Concepts and implementation in plasmo. jl. *Mathematical Programming Computation*, *14*(4), 699–747.

Ji, K., Yang, J., & Liang, Y. (2021). Bilevel optimization: Convergence analysis and enhanced design. *International Conference on Machine Learning*, 4882–4892.

Jiang, H., & Nachum, O. (2019,). *Identifying and Correcting Label Bias in Machine Learning*. https://arxiv.org/abs/1901.04966

Joachims, T. (1999). 11 Making Large-Scale Support Vector Machine Learning Practical. *Advances in Kernel Methods: Support Vector Learning*, 169.

Juditsky, A., & Nesterov, Y. (2014). Deterministic and stochastic primal-dual subgradient algorithms for uniformly convex minimization. *Stochastic Systems*, *4*(1), 44–80.

Kamani, M. M., Forsati, R., Wang, J. Z., & Mahdavi, M. (2021). Pareto efficient fairness in supervised learning: From extraction to tracing. *Arxiv Preprint Arxiv:2104.01634*.

Kamiran, F., & Calders, T. (2012). Data preprocessing techniques for classification without discrimination. *Knowledge and Information Systems*, *33*(1), 1–33.

Kerschke, P., Hoos, H. H., Neumann, F., & Trautmann, H. (2019). Automated algorithm selection: Survey and perspectives. *Evolutionary Computation*, *27*(1), 3–45.

Khaled, A., Mishchenko, K., & Jin, C. (2023). DoWG Unleashed: An Efficient Universal Parameter-Free Gradient Descent Method. *Advances in Neural Information Processing Systems*, *36*, 6748–6769.

Kim, I. Y., & Weck, O. de. (2004). Adaptive weighted sum method for multiobjective optimization. *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 4322.

Knueven, B., Ostrowski, J., & Watson, J.-P. (2020). On mixed-integer programming formulations for the unit commitment problem. *INFORMS Journal on Computing*, *32*(4), 857–876.

Knueven, B., Ostrowski, J., Castillo, A., & Watson, J.-P. (2022). A computationally efficient algorithm for computing convex hull prices. *Computers & Industrial Engineering*, *163*, 107806.

Koloskova, A., Hendrikx, H., & Stich, S. U. (2023). Revisiting gradient clipping: Stochastic bias and tight convergence guarantees. *International Conference on Machine Learning*, 17343–17363.

Kraft, D. (1994). Algorithm 733: TOMP–Fortran modules for optimal control calculations. *ACM Transactions on Mathematical Software (TOMS)*, *20*(3), 262–281.

Krasanakis, E., Spyromitros-Xioufis, E., Papadopoulos, S., & Kompatsiaris, Y. (2018). Adaptive sensitive reweighting to mitigate bias in fairness-aware classification. *Proceedings of the 2018 World Wide Web Conference*, 853–862.

Kuhn, H., & Tucker, A. (1951). Nonlinear programming. In J. Neyman (Ed.), *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability: Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*.

Kwon, J., Kwon, D., Wright, S., & Nowak, R. D. (2023). A fully first-order method for stochastic bilevel optimization. *International Conference on Machine Learning*, 18083–18113.

Lacoste-Julien, S., Schmidt, M., & Bach, F. (2012). A simpler approach to obtaining an O (1/t) convergence rate for the projected stochastic subgradient method. *Arxiv Preprint Arxiv:1212.2002*.

Lahoti, P., Gummadi, K. P., & Weikum, G. (2019). ifair: Learning individually fair data representations for algorithmic decision making. *2019 Ieee 35th International Conference on Data Engineering (Icde)*, 1334–1345.

Lemaréchal, C., Nemirovskii, A., & Nesterov, Y. (1995). New variants of bundle methods. *Mathematical Programming*, *69*(1), 111–147. https://doi.org/10.1007/BF01585555

Lessard, L., & Udell, M. (2025). Algebraic characterization of equivalence between optimization algorithms. *Arxiv Preprint Arxiv:2501.04972*.

Li, H., Linderman, G. C., Szlam, A., Stanton, K. P., Kluger, Y., & Tygert, M. (2017). Algorithm 971: An implementation of a randomized algorithm for principal component analysis. *ACM Transactions on Mathematical Software (TOMS)*, *43*(3), 1–14.

Li, T., & Lan, G. (2023,). *A simple uniformly optimal method without line search for convex optimization*.

Lin, Q., & Xiao, L. (2014). An Adaptive Accelerated Proximal Gradient Method and Its Homotopy Continuation for Sparse Optimization. *Proceedings of the 31st International Conference on Machine Learning, Proceedings of Machine Learning Research*, *32*, 73–81. https://proceedings.mlr.press/v32/lin14.html

Liu, S., & Vicente, L. N. (2022). Accuracy and fairness trade-offs in machine learning: A stochastic multi-objective approach. *Computational Management Science*, *19*(3), 513–537.

Löfberg, J. (2004). YALMIP: A toolbox for modeling and optimization in MATLAB. *2004 IEEE International Symposium on Computer Aided Control Systems Design*, 284–289. https://doi.org/10.1109/CACSD.2004.1393890

Lohia, P. K., Ramamurthy, K. N., Bhide, M., Saha, D., Varshney, K. R., & Puri, R. (2019). Bias mitigation post-processing for individual and group fairness. *Icassp 2019-2019 Ieee International Conference on Acoustics, Speech and Signal Processing (Icassp)*, 2847–2851.

Lu, S. (2023). Slm: A smoothed first-order lagrangian method for structured constrained nonconvex optimization. *Advances in Neural Information Processing Systems*, *36*, 80414–80454.

Lubin, M., Dowson, O., Garcia, J. D., Huchette, J., Legat, B., & Vielma, J. P. (2023). JuMP 1.0: Recent improvements to a modeling language for mathematical optimization. *Mathematical Programming Computation*, *15*(3), 581–589.

Maclaurin, D., Duvenaud, D., & Adams, R. (2015). Gradient-based hyperparameter optimization through reversible learning. *International Conference on Machine Learning*, 2113–2122.

Madani, M., & Van Vyve, M. (2017). A MIP framework for non-convex uniform price day-ahead electricity auctions. *EURO Journal on Computational Optimization*, *5*(1), 263–284.

Madani, M., Ruiz, C., Siddiqui, S., & Van Vyve, M. (2018,). *Convex Hull, IP and European Electricity Pricing in a European Power Exchanges setting with efficient computation of Convex Hull Prices*. arXiv.

Marler, R. T., & Arora, J. S. (2004). Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, *26*, 369–395.

Martinez, N., Bertran, M., & Sapiro, G. (2020). Minimax pareto fairness: A multi objective perspective. *International Conference on Machine Learning*, 6755–6764.

Mavrotas, G. (2009). Effective implementation of the \varepsilon-constraint method in multi-objective mathematical programming problems. *Applied Mathematics and Computation*, *213*(2), 455–465.

McMahan, B., Moore, E., Ramage, D., Hampson, S., & Arcas, B. A. y. (2017). Communication-efficient learning of deep networks from decentralized data. *Artificial Intelligence and Statistics*, 1273–1282.

Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., & Galstyan, A. (2021). A survey on bias and fairness in machine learning. *ACM Computing Surveys (CSUR)*, *54*(6), 1–35.

Menon, A. K., & Williamson, R. C. (2018). The cost of fairness in binary classification. *Conference on Fairness, Accountability and Transparency*, 107–118.

Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2012). *Foundations of machine learning*. MIT Press.

Mohri, M., Sivek, G., & Suresh, A. T. (2019). Agnostic federated learning. *International Conference on Machine Learning*, 4615–4625.

Moreau, T., Massias, M., Gramfort, A., Ablin, P., Bannier, P.-A., Charlier, B., Dagréou, M., Tour, T. Dupré la, Durif, G., F. Dantas, C., Klopfenstein, Q., Larsson, J., Lai, E., Lefort, T., Malézieux, B., Moufad, B., T. Nguyen, B., Rakotomamonjy, A., Ramzi, Z., … Vaiter, S. (2022,). Benchopt: Reproducible, efficient and collaborative optimization benchmarks. *Neurips*.

Nemirovskii, A. S. (1983). *Problem complexity and method efficiency in optimization*. Wiley.

Nesterov, Y. (2005). Smooth minimization of non-smooth functions. *Mathematical Programming*, *103*, 127–152.

Nesterov, Y. (2015). Universal gradient methods for convex optimization problems. *Mathematical Programming*, *152*(1–2), 381–404.

Nesterov, Y. (2004). *Introductory Lectures on Convex Optimization*. Springer US. https://doi.org/10.1007/978-1-4419-8853-9

Nesterov, Y. E. (1983). A method for solving the convex programming problem with convergence rate $O\left(\frac{1}{k^2}\right)$. *Doklady Akademii Nauk SSSR*, *269*(3), 543–547.

Obermeyer, Z., Powers, B., Vogeli, C., & Mullainathan, S. (2019). Dissecting racial bias in an algorithm used to manage the health of populations. *Science*, *366*(6464), 447–453.

Orabona, F., & Pál, D. (2016). Coin betting and parameter-free online learning. *Advances in Neural Information Processing Systems*, *29*.

Ozdayi, M. S., Kantarcioglu, M., & Iyer, R. (2021). Fair machine learning under limited demographically labeled data. *Arxiv Preprint Arxiv:2106.04757*.

Paquet, M. (2021). *Comparison between row generation, column generation and column-and-row generation for computing convex hull prices in day-ahead electricity markets* [Master's thesis]. http://hdl.handle.net/2078.1/thesis:30590

Pecci, F., & Jenkins, J. D. (2025). Regularized Benders Decomposition for High Performance Capacity Expansion Models. *IEEE Transactions on Power Systems*, *40*(4), 3105–3116. https://doi.org/10.1109/TPWRS.2025.3526413

Pedregosa, F. (2016). Hyperparameter optimization with approximate gradient. *International Conference on Machine Learning*, 737–746.

Pedregosa, F., Negiar, G., Askari, A., & Jaggi, M. (2020). Linearly convergent Frank-Wolfe with backtracking line-search. *International Conference on Artificial Intelligence and Statistics*, 1–10.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., VanderPlas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, *12*, 2825–2830.

Pillutla, K., Laguel, Y., Malick, J., & Harchaoui, Z. (2024). Federated learning with superquantile aggregation for heterogeneous data. *Machine Learning*, *113*(5), 2955–3022.

PJM Interconnection. (2017,). *Proposed enhancements to energy price formation*. Technical report.

Polyak, B. T. (1963). Gradient methods for the minimisation of functionals. *USSR Computational Mathematics and Mathematical Physics*, *3*(4), 864–878.

Rahimi, A., & Recht, B. (2007). Random features for large-scale kernel machines. *Advances in Neural Information Processing Systems*, *20*.

Ring, B. J. (1995). *Dispatch based pricing in decentralised power systems* [Doctoral dissertation].

Risch, R. H. (1969). The problem of integration in finite terms. *Transactions of the American Mathematical Society*, *139*, 167–189.

Rockafellar, R. T., & Uryasev, S. (2000). Optimization of conditional value-at-risk. *Journal of Risk*, *2*, 21–42.

Rockafellar, R. T. (1970). *Convex Analysis* (Issue 28). Princeton University Press.

Rockafellar, R. T. (1976). Monotone Operators and the Proximal Point Algorithm. *SIAM Journal on Control and Optimization*, *14*(5), 877–898. https://doi.org/10.1137/0314056

Roh, Y., Lee, K., Whang, S. E., & Suh, C. (2020). Fairbatch: Batch selection for model fairness. *Arxiv Preprint Arxiv:2012.01696*.

Rotaru, T., Glineur, F., & Patrinos, P. (2026). Exact worst-case convergence rates of gradient descent: a complete analysis for all constant stepsizes over nonconvex and convex functions: T. Rotaru et al. *Mathematical Programming*, 1–67.

Roulet, V., & d'Aspremont, A. (2017). Sharpness, restart and acceleration. *Advances in Neural Information Processing Systems*, *30*.

Rudi, A., Calandriello, D., Carratino, L., & Rosasco, L. (2018). On fast leverage score sampling and optimal learning. *Advances in Neural Information Processing Systems*, *31*.

Ruoss, A., Balunovic, M., Fischer, M., & Vechev, M. (2020). Learning certified individually fair representations. *Advances in Neural Information Processing Systems*, *33*, 7584–7596.

Ryu, N., & Min, S. (2019). Multiobjective optimization with an adaptive weight determination scheme using the concept of hyperplane. *International Journal for Numerical Methods in Engineering*, *118*(6), 303–319.

Sadykov, R., & Vanderbeck, F. (2013). Column generation for extended formulations. *EURO Journal on Computational Optimization*, *1*(1), 81–115. https://doi.org/10.1007/s13675-013-0009-9

Sagawa, S., Koh, P. W., Hashimoto, T. B., & Liang, P. (2019). Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *Arxiv Preprint Arxiv:1911.08731*.

Salzo, S., Villa, S., & others. (2012). Inexact and accelerated proximal point algorithms. *J. Convex Anal*, *19*(4), 1167–1192.

Scarf, H. E. (1994). The allocation of resources in the presence of indivisibilities. *Journal of Economic Perspectives*, *8*(4), 111–128.

Schmidt, M., Le Roux, N., & Bach, F. (2011). Convergence rates of inexact proximal-gradient methods for convex optimization. *Advances in Neural Information Processing Systems*, *24*.

Schölkopf, B., & Smola, A. J. (2001). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press. https://doi.org/10.7551/mitpress/4175.001.0001

Schölkopf, B., Herbrich, R., & Smola, A. J. (2001). A generalized representer theorem. *International Conference on Computational Learning Theory*, 416–426.

Shalev-Shwartz, S., & Zhang, T. (2013). Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, *14*, 567–599.

Shalev-Shwartz, S., Singer, Y., Srebro, N., & Cotter, A. (2011). Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical Programming*, *127*(1), 3–30.

Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel methods for pattern analysis*. Cambridge university press.

Shekhar, S., Fields, G., Ghavamzadeh, M., & Javidi, T. (2021). Adaptive sampling for minimax fair classification. *Advances in Neural Information Processing Systems*, *34*, 24535–24544.

Sivarajkumar, S., Huang, Y., & Wang, Y. (2023). Fair patient model: Mitigating bias in the patient representation learned from the electronic health records. *Journal of Biomedical Informatics*, *148*, 104544.

Steinwart, I., & Christmann, A. (2008). *Support vector machines*. Springer Science & Business Media.

Stevens, N., & Papavasiliou, A. (2022). Application of the Level Method for Computing Locational Convex Hull Prices. *IEEE Transactions on Power Systems*, *37*(5), 3958–3968.

Stevens, N., Papavasiliou, A., & Smeers, Y. (2024). On some advantages of convex hull pricing for the European electricity auction. *Energy Economics*, *134*, 107542. https://www.sciencedirect.com/science/article/pii/S0140988324002500

Streeter, M. (2023). Universal majorization-minimization algorithms. *Arxiv Preprint Arxiv:2308.00190*.

Tanji, S., Della Vecchia, A., Glineur, F., & Villa, S. (2023). Snacks: a fast large-scale kernel SVM solver. *2023 European Control Conference (ECC)*, 1–6.

Tanji, S., Kamri, Y., Glineur, F., & Madani, M. (2025). Dual first-order methods for efficient computation of convex hull prices. *Arxiv Preprint Arxiv:2504.01474*.

Tanji, S., Vaiter, S., & Laguel, Y. (2026). Fairness-informed Pareto Optimization: An Efficient Bilevel Framework. *Arxiv Preprint Arxiv:2601.13448*.

Taylor, A. B., Hendrickx, J. M., & Glineur, F. (2017c). Exact worst-case performance of first-order methods for composite convex optimization. *SIAM Journal on Optimization*, *27*(3), 1283–1313. https://doi.org/10.1137/16M108104X

Taylor, A. B., Hendrickx, J. M., & Glineur, F. (2017b). Performance Estimation Toolbox (PESTO): automated worst-case analysis of first-order optimization methods. *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 1278–1283. https://doi.org/10.1109/CDC.2017.8263832

Taylor, A. B., Hendrickx, J. M., & Glineur, F. (2017a). Smooth strongly convex interpolation and exact worst-case performance of first-order methods. *Mathematical Programming*, *161*(1), 307–345. https://doi.org/10.1007/s10107-016-1009-3

Tropp, J. A., Yurtsever, A., Udell, M., & Cevher, V. (2017). Fixed-rank approximation of a positive-semidefinite matrix from streaming data. *Advances in Neural Information Processing Systems*, *30*.

Tsang, M. Y., & Shehadeh, K. S. (2022). A Unified Framework for Analyzing and Optimizing a Class of Convex Fairness Measures. *Arxiv Preprint Arxiv:2211.13427*.

Tsang, M. Y., & Shehadeh, K. S. (2025). A Unified Framework for Analyzing and Optimizing a Class of Convex Fairness Measures. *Operations Research*.

Udell, M., Mohan, K., Zeng, D., Hong, J., Diamond, S., & Boyd, S. (2014). Convex optimization in Julia. *2014 First Workshop for High Performance Technical Computing in Dynamic Languages*, 18–28.

Vapnik, V. N. (1963). Pattern recognition using generalized portrait method. *Automation and Remote Control*, *24*(6), 774–780.

Vapnik, V. N. (1995,). *The nature of statistical learning theory*. Springer-Verlag.

Vapnik, V., & Vapnik, V. (1998). Statistical learning theory Wiley. *New York*, *1*(624), 2.

Villa, S., Salzo, S., Baldassarre, L., & Verri, A. (2013). Accelerated and inexact forward-backward algorithms. *SIAM Journal on Optimization*, *23*(3), 1607–1633.

Wächter, A., & Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, *106*(1), 25–57. https://doi.org/10.1007/s10107-004-0559-y

Wen, Z., Shi, J., Li, Q., He, B., & Chen, J. (2018). ThunderSVM: A fast SVM library on GPUs and CPUs. *The Journal of Machine Learning Research*, *19*(1), 797–801.

Williams, C., & Seeger, M. (2000). Using the Nyström method to speed up kernel machines. *Advances in Neural Information Processing Systems*, *13*.

Williamson, R., & Menon, A. (2019). Fairness risk measures. *International Conference on Machine Learning*, 6786–6797.

Xu, S., & Strohmer, T. (2023). Fair data representation for machine learning at the pareto frontier. *Journal of Machine Learning Research*, *24*(331), 1–63.

Xu, Y., Lin, Q., & Yang, T. (2019). Accelerate stochastic subgradient method by leveraging local growth condition. *Analysis and Applications*, *17*(05), 773–818.

Yang, T., Li, Y.-F., Mahdavi, M., Jin, R., & Zhou, Z.-H. (2012). Nyström method vs random fourier features: A theoretical and empirical comparison. *Advances in Neural Information Processing Systems*, *25*.

Yu, Y., Guan, Y., & Chen, Y. (2020). An extended integral unit commitment formulation and an iterative algorithm for convex hull pricing. *IEEE Transactions on Power Systems*, *35*(6), 4335–4346.

Zadeh, L. (1963). Optimality and non-scalar-valued performance criteria. *IEEE Transactions on Automatic Control*, *8*(1), 59–60.

Zafar, Valera, Gomez Rodriguez, et al. (2017). Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment. *Proceedings of the 26th International Conference on World Wide Web*, 1171–1180.

Zafar, M. B., Valera, I., Gomez Rodriguez, M., & Gummadi, K. P. (2017). Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment. *Proceedings of the 26th International Conference on World Wide Web*, 1171–1180.

Zafar, Valera, Gomez-Rodriguez, et al. (2019). Fairness constraints: A flexible approach for fair classification. *Journal of Machine Learning Research*, *20*(75), 1–42.

Zamani, M., & Glineur, F. (2023,). *Exact convergence rate of the last iterate in subgradient methods*.

Zemel, R., Wu, Y., Swersky, K., Pitassi, T., & Dwork, C. (2013). Learning fair representations. *International Conference on Machine Learning*, 325–333.

Zhang, H., Grossmann, I. E., McKinnon, K., Knudsen, B. R., Garcia Nava, R., & Tomasgard, A. (2025). Integrated investment, retrofit and abandonment energy system planning with multi-timescale uncertainty using stabilised adaptive Benders decomposition. *European Journal of Operational Research*, *325*, 261–280. https://doi.org/10.1016/j.ejor.2025.04.005

Zhang, H., Mazzi, N., McKinnon, K., Garcia Nava, R., & Tomasgard, A. (2024). A stabilised Benders decomposition with adaptive oracles for large-scale stochastic programming with short-term and long-term uncertainty. *Computers & Operations Research*, *167*, 106665. https://doi.org/10.1016/j.cor.2024.106665

Zhao, S., Lessard, L., & Udell, M. (2021). An automatic system to detect equivalence between iterative algorithms. *Arxiv Preprint Arxiv:2105.04684*.